The Pennsylvania State University

The Graduate School

College of Engineering

ROBUST, ADAPTABLE MANY-OBJECTIVE OPTIMIZATION: THE FOUNDATIONS, PARALLELIZATION AND APPLICATION OF THE BORG MOEA

A Dissertation in

Computer Science and Engineering

by

David M. Hadka

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2013

The dissertation of David M. Hadka was received and approved* by the following:

Kamesh Madduri
Assistant Professor of Computer Science and Engineering
Dissertation Co-Adviser
Chair of Committee

Patrick Reed
Associate Professor of Civil and Environmental Engineering
Dissertation Co-Adviser

Sofya Raskhodnikova
Associate Professor of Computer Science and Engineering

Soundar R. T. Kumara
Allen E. & M. Pearce Professor of Industrial and Manufacturing Engineering
Affiliated Professor of Computer Science and Engineering

Timothy W. Simpson
Professor of Mechanical Engineering and Industrial Engineering

Mark T. Traband
Research Associate at the Applied Research Laboratory
Affiliate Faculty of Industrial and Manufacturing Engineering

Lee Coraor
Director of Academic Affairs, Computer Science and Engineering

*Signatures are on file in the Graduate School.

ABSTRACT

This dissertation presents the design, development, and parallelization of the Borg Multiobjective Evolutionary Algorithm (MOEA), an efficient and robust many-objective optimization tool. It is characterized by its use of auto-adaptive multi-operator search and other adaptive features, allowing the algorithm to tailor itself to local search conditions encountered during optimization. Using a rigorous diagnostic framework, the Borg MOEA is distinguished against a broad sample of state-of-the-art MOEAs. The Borg MOEA meets or exceeds the efficiency, reliability, and search quality of other MOEAs on the majority of tested problems. To extend the application of the Borg MOEA to time-consuming, complex engineered systems, we develop two parallel versions of the Borg MOEA: (1) the master-slave and (2) the multi-master Borg MOEA. These parallel versions are capable of running efficiently on large-scale computing systems, exploiting tens of thousands of processors. Such large-scale computing allows the Borg MOEA to optimize complex engineered systems efficiently while producing high-quality results with high reliability. This work culminates with two real-world case studies of complex engineered systems: (1) the General Aviation Aircraft (GAA) design problem and (2) a risk-based water supply portfolio planning problem.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Daniel Finke, Chris Ligetti, Mark Traband, and my other colleagues at The Pennsylvania State University Applied Research Laboratory for their support and encouragement over the years. I would also like to thank Joshua Kollat, Joseph Kasprzyk, Rachel Urban, Alisha Fernandez, Matthew Woodruff, Jonathan Herman, Martha Butler, and Ruchit Shah for their support of and contributions to this work. Thanks to my committee members for reviewing this dissertation and providing helpful feedback. Last but not least, special thanks to Patrick Reed for nurturing my interest in this subject matter and helping guide this research.

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this dissertation.

This work was supported in part through instrumentation funded by the National Science Foundation through grant OCI-0821527.

It is a mistake to think you can solve any major problem with just potatoes.
- Douglas Adams

Chapter 1

Introduction

Multiobjective evolutionary algorithms (MOEAs) are a class of optimization algorithms inspired by the processes of natural evolution (Holland, 1975). As early as 1984, researchers have been interested in solving problems with multiple conflicting objectives using evolutionary algorithms (Schaffer, 1984). Since then, researchers have successfully applied MOEAs to a large array of problems from industrial, electrical, computer, civil and environmental engineering; aeronautics; finance; chemistry; medicine; physics; and computer science (for a detailed overview see Coello Coello et al. (2007)).

In early MOEA research, only a small number of algorithms existed to solve multiobjective optimization problems and their performance was limited to fairly simple two and three objective formulations. It was the introduction of the Nondominated Sorting Genetic Algorithm-II (NSGA-II) (Deb et al., 2000) that revolutionized the field, dramatically increasing the use of Pareto dominance-based optimization in an extremely diverse array of applications (Coello Coello et al., 2007). Even with the large number of MOEAs developed after NSGA-II, it remains one of the most widely used and cited MOEAs to date.

In recent years, burgeoning computing power and an increasing acceptance of MOEAs as multiobjective optimizers has led researchers to solve problems with four or more objectives (Fleming et al., 2005; Coello Coello et al., 2007; di Pierro et al., 2007; Ferringer et al., 2009; Kasprzyk et al., 2009). Quickly, however, a number of theoretical and experimental issues were observed. Such problems were termed *many-objective* and were shown to strain traditional MOEAs, which were originally designed for only two or three objectives. In some cases, complete search failures were observed on many-objective problems (Purshouse and Fleming, 2003, 2007).

A quick thought exercise gives light to the complexities that arise during many-objective optimization. Consider a 6 objective problem. An MOEA solving this problem must consider all interactions and tradeoffs among the 6 objectives. However, encoded in the 6 objective formulation are all subproblems of lower dimension. This includes 6 single-objective subproblems, 15 two-objective subproblems, 20 three-objective subproblems, 15 four-objective subproblems and 6 five-objective subproblems. In total, 63 subproblems must be simultaneously solved by the MOEA.

One early and popular approach for handling many objectives involves aggregating the

many objectives into a single objective. This single objective is subsequently solved using a single-objective evolutionary algorithm (EA). Many practical problems exist in this approach, however, that limit its applicability. Not only will aggregating the objectives hide the complex interactions and tradeoffs among the many objectives, Kasprzyk et al. (2009) and Kollat et al. (2011) demonstrate that it is necessary to use the full many-objective formulation to avoid myopic decision making. For instance, Kasprzyk et al. (2009) demonstrated that using lower-dimensional formulations lead to severe decision errors in a water portfolio planning system, increasing the potential for costly failures when planning a city's water portfolio.

Hence, there is significant interest in solving the full many-objective formulations of complex, real-world problems. This approach, unfortunately, is not without its own set of issues. Farina and Amato (2004), Fleming et al. (2005), and Purshouse and Fleming (2007) observed that the proportion of locally Pareto non-dominated solutions tends to become large as the number of objectives increases. This is a direct result of Pareto dominance and its aim to capture, without preference, the entire tradeoff surface between two or more conflicting objectives. This leads to difficulties in producing offspring that dominate poorly performing, but still non-dominated, members in the population — a phenomenon termed *dominance resistance* (Hanne, 2001; Ikeda et al., 2001; Purshouse and Fleming, 2007). This increasing proportion of locally Pareto non-dominated solutions and the phenomenon of dominance resistance can impact the performance of MOEAs in several ways.

First, these conditions may limit the ability of dominance relations (e.g., Pareto dominance) in differentiating high-quality and low-quality solutions. Several researchers have proposed alternate dominance relations to provide more stringent dominance criteria. One must, however, be aware of the impact of selecting a different dominance relation, as it may focus search towards a subspace and fail to produce solutions along the entire extent of the tradeoff surface (Coello Coello et al., 2007).

Second, as the proportion of locally Pareto non-dominated solutions increases and the offspring are likely to also be non-dominated as a result of dominance resistance, it is often difficult for an MOEA to identify which offspring should survive and replace existing members in the population. In such scenarios, the diversity operator, such as crowding, is often the primary mechanism for determining survival. This phenomenon is termed *active diversity maintenance* (Purshouse and Fleming, 2007).

Third, Hanne (1999) observed that active diversity maintenance can cause *deterioration*. Deterioration occurs whenever the solution set discovered by an MOEA at time i contains one or more solutions dominated by a solution discovered at some earlier point in time $j < i$. In the extreme, deterioration can cause an MOEA to diverge away from the Pareto front. Laumanns et al. (2002) effectively eliminate deterioration with the ϵ-dominance relation; however, at present, most state-of-the-art MOEAs in use today have yet to adopt mechanisms for avoiding deterioration.

Lastly, Teytaud (2006, 2007) show that dominance resistance can cause the convergence rate of MOEAs to degrade to be no better than random search for problems with ten or more objectives. This result is backed by Ishibuchi et al. (2008a), where it is demonstrated

that several state-of-the-art MOEAs fail on problems with as few as four objectives.

Clearly, there exists significant interest in many-objective optimization, but key innovations are necessary in order to overcome these documented problems. This dissertation documents a sequence of studies to better understand and develop the theory and foundations for robust many-objective optimization. First, a novel MOEA was designed specifically for handling complex, many-objective problems where the primary future focus will be on advancing severely challenging real-world applications. In order to facilitate these design goals, the proposed Borg MOEA assimilates several design principles from existing MOEAs and introduces several novel components. These components include:

1. an ϵ-box dominance archive for maintaining convergence and diversity throughout search (Laumanns et al., 2002);

2. ϵ-progress, which is a computationally efficient measure of search progression and stagnation;

3. an adaptive population sizing operator based on ϵ-NSGA-II's (Kollat and Reed, 2006) use of time continuation to maintain search diversity and to facilitate escape from local optima;

4. multiple recombination operators to enhance search across a wide assortment of problem domains; and

5. the steady-state, elitist model of ϵ-MOEA (Deb et al., 2003), which can be easily extended for use on parallel architectures.

Next, a comprehensive comparative study between the Borg MOEA and a number of competing MOEAs was conducted. Not only does this study test more algorithms and problems than previously attempted in the literature, it proposes new performance measures for differentiating the quality, reliability and efficiency of the tested MOEAs. In addition, Sobol' global variance decomposition is used to decompose the relative importance of and interactions among each algorithm's parameters. By identifying key parameters and their complex interactions, guidance on parameterizing and controlling the algorithms can be inferred.

Lastly, in order to facilitate large-scale, time-consuming problems, two parallel variants of the Borg MOEA were developed. Not only do the parallel variants significantly reduce the time needed to solve such problems, they drastically improve the overall search quality. Discrete event simulation results detailing the necessary conditions to maximize speedup and efficiency were developed and incorporated to maximize the potential of the parallel variants. Both variants were applied to a severely constrained, many-objective complex engineered system.

The result of this dissertation research includes the serial and parallel versions of the Borg MOEA, and the theoretical and experimental results to support these claims. The remainder of this dissertation is organized as follows.

Chapter 2 - Background
This chapter provides the reader with sufficient background to understand the primary components of this dissertation. It formally defines multiobjective optimization, Pareto optimality and many-objective optimization. The theoretical and experimental issues observed in the literature concerning many-objective optimization are detailed. Chapter 2 concludes with descriptions of the MOEAs and test problems used throughout this dissertation.

Chapter 3 - Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework
Chapter 3 introduces the Borg MOEA for many-objective optimization. The Borg MOEA combines ϵ-dominance, a measure of convergence speed named ϵ-progress, randomized restarts and auto-adaptive multioperator recombination into a unified optimization framework. A comparative study on 33 instances of 18 test problems from the DTLZ (Deb et al., 2001), WFG (Huband et al., 2006), and CEC 2009 (Zhang et al., 2009b) test suites demonstrates that the Borg MOEA meets or exceeds 6 state-of-the-art MOEAs on the majority of the tested problems.

Chapter 4 - Framework for Diagnosing Search Controls and Failure Modes
Extending the comparative study from the previous chapter, Chapter 4 introduces a diagnostic framework for rigorously assessing the search controls and failure modes of MOEAs. Using this methodology, it is possible to carefully determine an MOEA's search quality, reliability, efficiency and controllability. Applying this framework to the Borg MOEA and 8 other state-of-the-art many-objective optimizers solidifies the contributions provided by the Borg MOEA. This study represents the most comprehensive evaluation of the state-of-the-field ever completed.

Chapter 5 - Case Study: Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems
Chapter 5 explores the application of the Borg MOEA on a real-world product family design problem: the severely constrained, ten objective General Aviation Aircraft (GAA) problem. The GAA problem represents a promising benchmark problem that strongly highlights the importance of using auto-adaptive search to discover how to exploit multiple recombination strategies cooperatively. The auto-adaptive behavior of the Borg MOEA is rigorously compared against its ancestor algorithm, the ϵ-MOEA, by employing global sensitivity analysis across each of the algorithm's feasible parameter ranges. This provides the first application of the Sobol' sensitivity analysis from Chapter 4 to determine the individual and interactive parameter sensitivities of MOEAs on a real-world many-objective problem.

Chapter 6 - Large-Scale Parallelization of the Borg MOEA
The previous chapters have identified the number of function evaluations (NFE) as the key controlling parameter of the Borg MOEA. Therefore, it is logical to build a parallel implementation of the Borg MOEA to increase NFE by running on large-scale computing

systems. Chapter 6 develops two parallel variants of the Borg MOEA. The master-slave Borg MOEA is designed to scale to thousands of processors. The multi-master Borg MOEA is designed to scale on emerging Petascale systems. Both parallel variants retain the auto-adaptive features of the serial Borg MOEA from Chapter 3 but also introduce several features designed to improve the reliability of the algorithm on large, complex, severely constrained problems.

Chapter 7 - Scalability of the Parallel Borg MOEA
Chapter 7 provides a preliminary exploration of the scalability of the two parallel Borg MOEA implementations on the 5D DTLZ2 and UF11 test problems. This chapter starts with an experimental comparison of the parallel efficiency and hypervolume speedup of the master-slave and multi-master Borg MOEA. Next, models for predicting the runtime, efficiency, and lower and upper processor bounds are derived. This includes a discrete event simulation model for accurately modeling the complex interactions in the parallel Borg MOEA. Lastly, these models are used to provide guidance for optimally configuring the parallel Borg MOEA.

Chapter 8 - Case Study: Risk-Based Water Supply Portfolio Planning
Chapter 8 explores the application of the parallel Borg MOEA on a real-world complex engineered system: a severely constrained, six objective risk-based water supply portfolio planning problem. This problem features many of the challenging problem properties discussed in Chapter 6. It is many-objective, multi-modal, non-linear, contains a mix of discrete and real decision variables, is severely constrained, and has stochastic objectives with expensive function evaluation times. Using this problem, we demonstrate that the parallel variants of the Borg MOEA developed in Chapter 6 significantly improve speed of convergence, solution quality, and reliability.

Chapter 9 - Conclusions, Contributions, and Future Work
Chapter 9 concludes this dissertation by detailing the results from the prior chapters and discussing the impact the Borg MOEA has on solving large-scale, complex engineered systems. Additionally, the contributions to the fields of parallel computing, evolutionary computation, and operations research resulting from this dissertation are outlined. Lastly, future research directions that can extend and improve the work presented in this dissertation are proposed.

Chapter 2

Background

This chapter provides introductions to core technical concepts that are utilized throughout this dissertation, and provides a more detailed review of historical work focused on multiobjective evolutionary algorithms (MOEAs). Section 2.1 introduces the goals of multiobjective optimization. Section 2.2 formally defines the multiobjective problem class. Section 2.3 introduces the concept of Pareto optimality, which captures the notion of optimality for multiobjective problems. Section 2.4 presents the motivation behind MOEAs. Section 2.5 discusses the extension of MOEAs to many-objective problems, which are problems with ≥ 4 objectives. Section 2.6 and Section 2.7 detail the MOEAs and test problems used for testing throughout this dissertation. Lastly, Section 2.8 discusses the techniques to measure solution quality when testing MOEAs.

2.1 Multiobjective Optimization

Optimization is the process of identifying the best solution among a set of alternatives (Miettinen, 1999). Whereas single objective optimization employs a single criterion for identifying the best solution among a set of alternatives, multiobjective optimization employs two or more criteria. The criteria used to compare solutions are known as *objectives*. As multiple objectives can conflict with one another — i.e., improving one objective leads to the deterioration of another — there is, generally speaking, no single optimal solution to multiobjective problems.

As an example, Figure 2.1 shows the tradeoff between two objectives: (1) cost and (2) error. The shaded region depicts the set of candidate solutions to this hypothetical problem. The top-left region contains low cost, high error candidate solutions. The bottom-right region contains high cost, low error candidate solutions. Between these two extremes lie the various degrees of tradeoff between the two objectives, where increases in cost lead to reduced error.

Figure 2.1 demonstrates a fundamental issue in multiobjective optimization. Given that there is no single optimal solution, rather a multitude of potential solutions with varying degrees of tradeoff between the objectives, decision-makers are subsequently responsible for

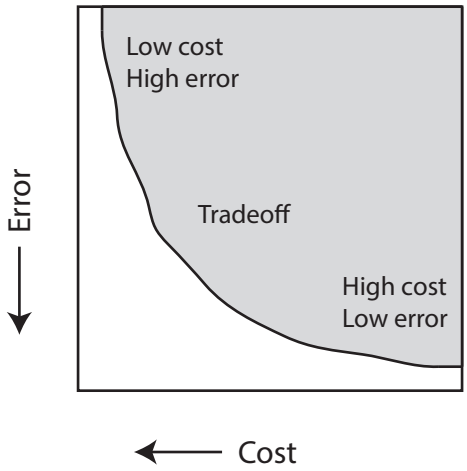


Figure 2.1: Example of the tradeoff between two objectives: (1) cost and (2) error. A tradeoff is formed between these two conflicting objectives where increases in cost lead to reduced error. All figures in this dissertation showing objectives include arrows pointing towards the ideal optimum.

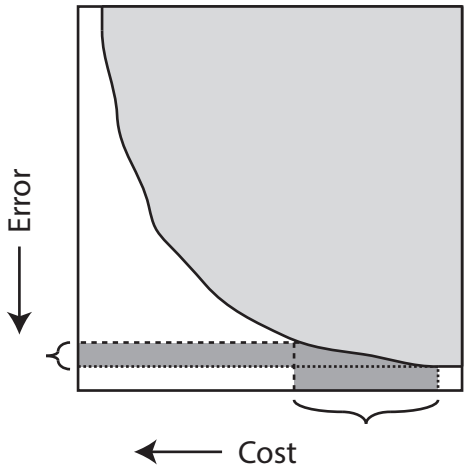


Figure 2.2: Example showing the effect of diminishing returns, where a large increase in cost is necessary to impart a marginal reduction in error.

exploring this set of potential solutions and identifying the solution(s) to be implemented. While ultimately the selection of the final solution is the responsibility of the decision-maker, optimization tools should assist this decision process to the best of their ability. For instance, it may prove useful to identify points of diminishing returns. For example, Figure 2.2 identifies the region where a large increase in cost is necessary to impart a marginal decrease in error. To perform this type of analysis, it is necessary to provide the decision-maker with an enumeration or approximation of these tradeoffs. This strategy of enumerating or approximating the tradeoffs is known as *a posteriori* optimization (Coello Coello et al., 2007), and is the focus of this dissertation.

2.2 Multiobjective Problem

A multiobjective problem (MOP) with M objectives is defined as

$$\begin{aligned} \underset{\mathbf{x}\in\Omega}{\text{minimize}} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ \text{subject to} \quad & c_i(\mathbf{x}) = 0,\ \forall i \in \mathcal{E}, \\ & c_j(\mathbf{x}) \leq 0,\ \forall j \in \mathcal{I}. \end{aligned} \tag{2.1}$$

We call $\mathbf{x}$ the *decision variables*, which is the vector of variables that are manipulated during the optimization process:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_L \end{bmatrix} \tag{2.2}$$

Decision variables can be defined in a variety of ways, but it is common to see the following types (Bäck et al., 1997):

- **Real-Valued:** `0.1134`, with optional lower and upper bounds
- **Binary:** `00110001001010000101111010110111001 1`
- **Permutation:** `4,2,0,1,3`

In some applications, it is possible for the number of decision variables, L, to not be a fixed value. In this dissertation, however, we assume that L is constant for a given problem.

The decision space, Ω, is the set of all decision variables. The MOP may impose restrictions on the decision space, called *constraints*. As an example, in Figure 2.3, a hypothetical constraint would prevent any solutions from exceeding an error threshold. In this manner, constraints inform the optimization process as to which solutions are infeasible or impractical. Equation (2.1) shows that zero or more constraints $c_i(\mathbf{x})$ can be defined to express both equality and inequality constraints. The sets $\mathcal{E}$ and $\mathcal{I}$ define whether the constraint is an equality or inequality constraint. The set of all decision variables in Ω which are feasible (i.e., satisfy all constraints) define the *feasible region*, Λ.

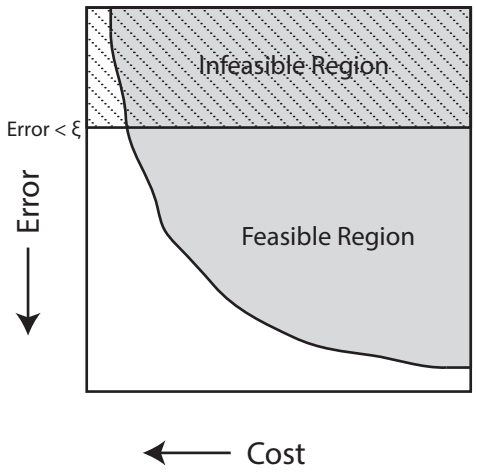


Figure 2.3: Example showing how constraints define an infeasible region (the hashed region). Valid solutions to the optimization problem are only found in the feasible region.

2.3 Pareto Optimality

The notion of optimality used today is adopted from the work of Francis Ysidro Edgeworth and Vilfredo Pareto (Coello Coello et al., 2007), and is commonly referred to as *Pareto optimality*. Pareto optimality considers solutions to be superior or inferior to another solution only when it is superior in all objectives or inferior in all objectives, respectively. The tradeoffs in an MOP are captured by solutions which are superior in some objectives but inferior in others. Such pairs of solutions which are both superior and inferior with respect to certain objectives are called *non-dominated*, as shown in Figure 2.4. More formally, the notion of Pareto optimality is defined by the Pareto dominance relation:

Definition 1. *A vector* $\mathbf{u} = (u_1, u_2, \ldots, u_M)$ ***Pareto dominates*** *another vector* $\mathbf{v} = (v_1, v_2, \ldots, v_M)$ *if and only if* $\forall i \in \{1, 2, \ldots, M\}, u_i \leq v_i$ *and* $\exists j \in \{1, 2, \ldots, M\}, u_j < v_j$. *This is denoted by* $\mathbf{u} \prec \mathbf{v}$.

Two solutions are non-dominated if neither Pareto dominates the other (i.e., $\mathbf{u} \not\prec \mathbf{v}$ and $\mathbf{v} \not\prec \mathbf{u}$). The set of all non-dominated solutions is captured by the Pareto optimal set and the Pareto front. The former contains the decision variables while the latter contains the objectives.

Definition 2. *For a given multiobjective problem, the* ***Pareto optimal set*** *is defined by*

$$\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \neg\exists \mathbf{x}' \in \Lambda, F(\mathbf{x}') \prec F(\mathbf{x})\}$$

Definition 3. *For a given multiobjective problem with Pareto optimal set* $\mathcal{P}^*$, *the* ***Pareto front*** *is defined by*

$$\mathcal{PF}^* = \{F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\}$$

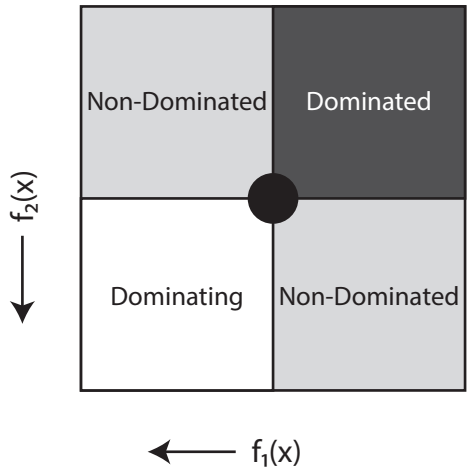


Figure 2.4: Depiction of the various Pareto dominance regions. These regions are relative to each solution, which is centered in the figure. The *dominated* region is inferior in all objectives, the *dominating* region is superior in all objectives and the *non-dominated* region is superior in one objective but inferior in the other.

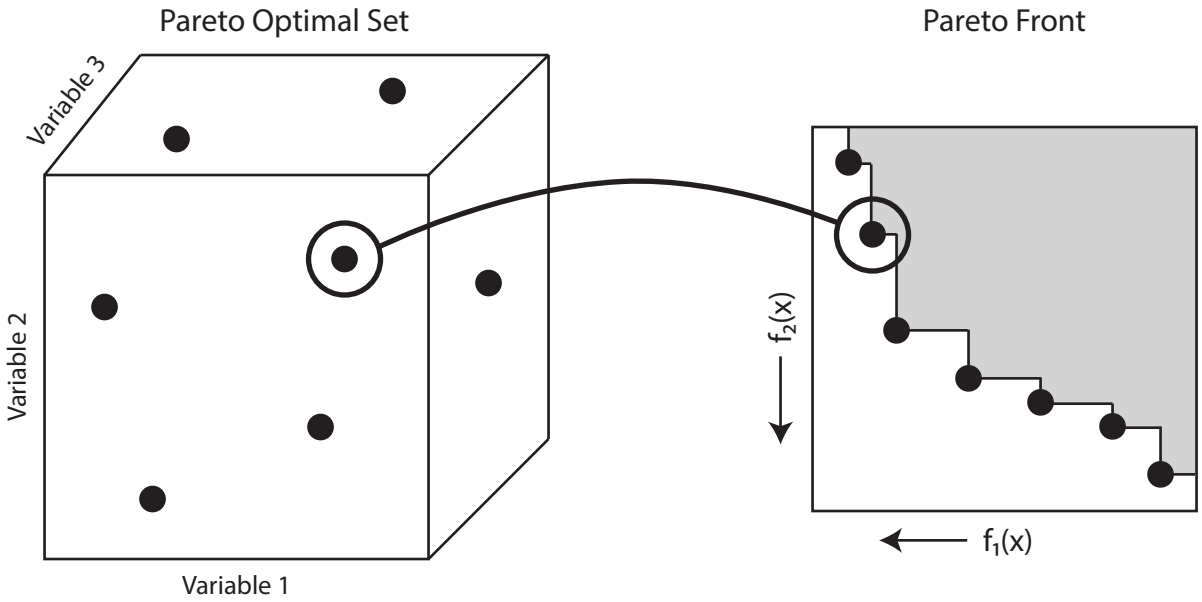


Figure 2.5: Shows a hypothetical mapping between a 3-dimensional Pareto optimal set and its associated 2-dimensional Pareto front. The shaded region in the Pareto front shows the space dominated by the Pareto front.

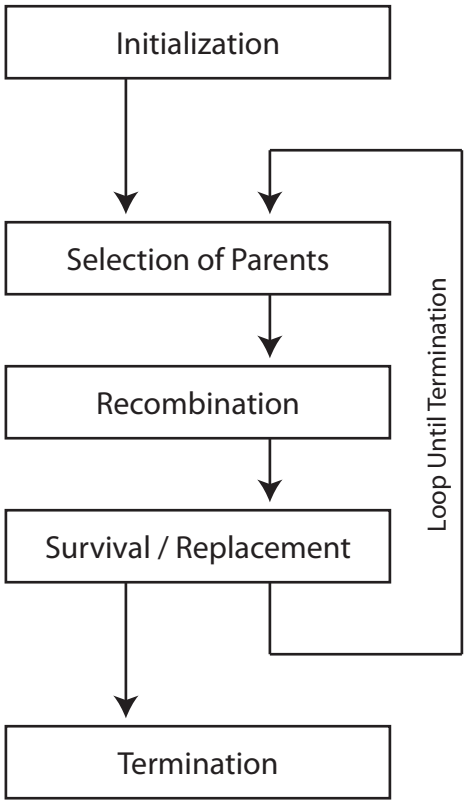


Figure 2.6: The outline of a simple EA. EAs begin with an initialization process, where the initial search population is generated. They next enter a loop of selecting parent individuals from the search population, applying a recombination operator (such as crossover and mutation in genetic algorithms) to generate offspring, and finally updating the search population with these offspring using a replacement strategy. This loop is repeated until some termination condition is met, usually after a fixed number of objective function evaluations (NFE). Upon termination, the EA reports the set of optimal solutions discovered during search.

In this dissertation, the Pareto dominance relation is applied to the objectives. For convenience, we use $\mathbf{x} \prec \mathbf{y}$ interchangeably with $F(\mathbf{x}) \prec F(\mathbf{y})$.

Figure 2.5 shows an example Pareto optimal set and Pareto front, and the resulting mapping between the two. The Pareto optimal set defines the decision variables, whereas the Pareto front captures the objectives and their tradeoffs via Pareto optimality.

2.4 Multiobjective Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of search and optimization algorithms inspired by processes of natural evolution (Holland, 1975). A broad overview of the design and development of EAs is provided in Bäck et al. (1997). The outline of a simple EA is shown in Figure 2.6. EAs begin with an initialization process, where the initial search population is

generated. They next enter a loop of selecting parent individuals from the search population, applying a recombination operator to generate offspring, and finally updating the search population with these offspring using a replacement strategy. This loop is repeated until some termination condition is met, usually after a fixed number of objective function evaluations (NFE). Upon termination, the EA reports the set of optimal solutions discovered during search.

The behavior of the selection, recombination and survival/replacement processes typically depend on the "class" of EA. For instance, genetic algorithms (GAs) apply crossover and mutation operators that mimic genetic reproduction via DNA (Holland, 1975). Particle swarm optimization (PSO) algorithms simulate flocking behavior, where the direction of travel of each individual is steered towards the direction of travel of surrounding individuals (Kennedy and Eberhart, 1995). While the behavior of each class may be vastly different, they all share a common attribute of utilizing a search population.

Their ability to maintain a population of diverse solutions makes EAs a natural choice for solving MOPs. Early attempts at solving MOPs involved using aggregation-based approaches (Bäck et al., 1997). In aggregation-based approaches, the decision-maker defines an aggregate fitness function that transforms the MOP into a single objective problem, which can subsequently be solved with an EA. Two commonly-used aggregate fitness functions are linear weighting:

$$F_L(\mathbf{x}) = \sum_{i=1}^{M} \lambda_i f_i(\mathbf{x}), \tag{2.3}$$

and the weighted Chebyshev method:

$$F_T(\mathbf{x}) = \max_{i=1,2,\ldots,M} \left(\lambda_i \left| z_i^* - f_i(\mathbf{x}) \right|\right), \tag{2.4}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_M)$ are the weights and $\mathbf{z}^* = (z_1^*, z_2^*, \ldots, z_M^*)$ is a reference point identifying the decision-maker's goal (note: this reference point need not be a feasible solution).

Coello Coello et al. (2007) discusses the advantages and disadvantages of aggregate fitness approaches. The primary advantage is the simplicity of the approach and the ability to exploit existing EAs to solve MOPs. In addition, appropriately defined aggregate fitness functions can provide very good approximations of the Pareto front. However, poorly-weighted aggregate fitness functions may be unable to find non-convex regions of the Pareto front. This is problematic since selecting appropriate weights is non-trivial, especially if the relative worth of each objective is unknown or difficult to quantify. Lastly, in order to generate multiple Pareto optimal solutions, aggregate fitness approaches need to be run with differing weights to generate solutions across the entire Pareto front.

These limitations lead to the development of multiobjective evolutionary algorithms (MOEAs) that search for multiple Pareto optimal solutions in a single run. The first MOEA to search for multiple Pareto optimal solutions, the Vector Evaluated Genetic Algorithm (VEGA), was introduced by Schaffer (1984). VEGA was found to have problems similar to aggregation-based approaches, such as an inability to generate concave regions of the

Pareto front. Goldberg (1989a) was first to suggest the use of Pareto-based selection, but this concept was not applied until 1993 in the Multiobjective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993). Between 1993 and 2003, several *first-generation* MOEAs were introduced demonstrating important design concepts such as elitism, diversity maintenance and external archiving. Notable first-generation algorithms include the Niched-Pareto Genetic Algorithm (NPGA) (Horn and Nafpliotis, 1993), the Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994), the Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999), the Pareto-Envelope based Selection Algorithm (PESA) (Corne and Knowles, 2000) and the Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 1999). Many of these MOEAs have since been revised to incorporate more efficient algorithms and improved design concepts. To date, Pareto-based approaches outnumber aggregate fitness approaches (Coello Coello et al., 2007). For a more comprehensive overview of the historical development of MOEAs, please refer to the text by Coello Coello et al. (2007).

2.5 Many-Objective Optimization

In the past twenty years, researchers have successfully applied MOEAs to a large array of problems from industrial, electrical, computer, civil and environmental engineering; aeronautics; finance; chemistry; medicine; physics and computer science (Coello Coello et al., 2007). While in the majority of these domains MOEAs have been used predominately to solve two or three objective problems, there are growing demands for addressing higher-dimensional problems. This has lead to a growing research community in *many-objective optimization* (Fleming et al., 2005; Adra and Fleming, 2009).

While many-objective applications are growing in their success, there exists strong theoretical and experimental evidence suggesting that existing approaches are insufficient for many-objective problems. Farina and Amato (2004), Fleming et al. (2005) and Purshouse and Fleming (2007) observe that the proportion of locally Pareto non-dominated solutions tends to become large as the number of objectives increases. This is a direct result of Pareto dominance and its aim to capture, without preference, the entire tradeoff surface between two or more objectives. This leads to difficulties in producing offspring that dominate poorly performing, but still non-dominated, members in the population — a phenomenon termed *dominance resistance* (Hanne, 2001; Ikeda et al., 2001; Purshouse and Fleming, 2007). This increasing proportion of locally Pareto non-dominated solutions and the phenomenon of dominance resistance can impact the performance of MOEAs in several ways.

First, these conditions may limit the ability of dominance relations in differentiating high-quality and low-quality solutions. Several researchers have proposed alternate dominance relations to provide more stringent dominance criteria, including the preferability (Fonseca and Fleming, 1998), preferred (Drechsler et al., 2001), ϵ-preferred (Sülflow et al., 2007), k-optimality (Farina and Amato, 2004) and preference order ranking (di Pierro et al., 2007) dominance relations. Corne and Knowles (2007) propose using classical methods to rank non-dominated objective vectors, such as *average ranking*, which have been shown to provide

competitive results. One must, however, be aware of the impact of selecting a different dominance relation, as it may focus search towards a subspace and fail to produce solutions along the entire extent of the tradeoff surface (Coello Coello et al., 2007).

Second, as the proportion of locally Pareto non-dominated solutions increases and the offspring are likely to also be non-dominated as a result of dominance resistance, it is often difficult for an MOEA to identify which offspring should survive and replace existing members in the population. In such scenarios, the diversity operator, such as crowding, is often the primary mechanism for determining survival. This phenomenon is termed *active diversity maintenance* (Purshouse and Fleming, 2007).

Third, Hanne (1999) observed that active diversity maintenance can cause *deterioration*. Deterioration occurs whenever the solution set discovered by an MOEA at time i contains one or more solutions dominated by a solution discovered at some earlier point in time $j < i$. In the extreme, deterioration can cause an MOEA to diverge away from the Pareto front. Laumanns et al. (2002) effectively eliminate deterioration with the ϵ-dominance relation; however, at present, most state-of-the-art MOEAs in use today have yet to adopt mechanisms for avoiding deterioration.

Lastly, as detailed in Chapter 4, we show empirically on several MOEAs that parameterization can greatly impact the performance of an MOEA. For many top-performing algorithms, proper parameterization becomes severely challenging as the number of objectives increases. In addition, we demonstrate that most modern MOEAs can fail in terms of both convergence and reliability on test problems with as few as four objectives. These results are backed by the theoretical work of Teytaud (2006, 2007), which show that dominance resistance can cause the convergence rate of MOEAs to degrade to be no better than random search for problems with ten or more objectives, and the experimental work of Ishibuchi et al. (2008a), where it is also demonstrated that several state-of-the-art MOEAs fail on problems with as few as four objectives.

A variety of methods have been proposed in the literature for addressing many-objective optimization. The following briefly overviews the most common methods.

Aggregate Fitness Functions Using aggregation functions to convert a multiobjective problem into a single-objective problem have remained popular, but special care must be taken when designing the aggregation function to avoid its potential pitfalls (Wagner et al., 2007). However, it is suggested in the literature that aggregate fitness functions may be particularly advantageous on many-objective problems since they avoid scaling issues (Ishibuchi et al., 2009). However, this claim has yet to be sufficiently demonstrated across a variety of challenging many-objective problems.

Indicator-Based Methods Indicator-based methods replace the Pareto dominance relation with an indicator function intended to guide search towards regions of interest (Ishibuchi et al., 2010). The hypervolume measure is often used as the indicator function due to its theoretical characteristics (Ishibuchi et al., 2010). Hypervolume-based methods avoid active diversity maintenance by not using an explicit diversity-preserving mechanisms, and instead

promote diversity through the hypervolume measure itself (Wagner et al., 2007). One potential downfall to hypervolume-based methods is the computational complexity of calculating the hypervolume measure on high-dimensional problems, but Ishibuchi et al. (2010) have proposed an approximation method to reduce the computational complexity.

Pareto Front Approximation Issues like deterioration arise when finite population sizes force an MOEA to remove Pareto non-dominated solutions during replacement (Laumanns et al., 2002). Excessive deterioration can cause the MOEA to diverge away from the Pareto front. As the proportion of Pareto non-dominated solutions increases as the number of objectives increases, the occurrence of deterioration increases. Laumanns et al. (2002) introduced the ϵ-dominance relation as a way to eliminate deterioration by approximating the Pareto front, and also provided theoretical proofs of convergence and diversity for algorithms using this relation (if the algorithm satisfies several additional necessary conditions).

Space Partitioning and Dimensionality Reduction Both space partitioning and dimensionality reduction methods attempt to convert many-objective problems into lower-dimensional instances that can be solved effectively using existing MOEAs. Space partitioning methods attempt to emphasize search in lower-dimensional objective spaces by partitioning the M-objective space of the original problem into many disjoint lower-dimensional subspaces, each of which is searched independently (Aguirre and Tanaka, 2009). On the other hand, dimensionality reduction methods attempt to convert the higher-dimensional objective space into a lower-dimensional representation using methods like principal component analysis (PCA) (Saxena and Deb, 2008).

Rotational Invariance While not specifically a method for many-objective optimization, the importance of rotational invariance is only briefly explored in the literature but its impacts on real-world problems are significant (Coello Coello et al., 2007; Iorio and Li, 2008). Rotational invariance relates to the effects of conditional dependencies between decision variables and the recombination operators. In unrotated problems (i.e., decision variables are independent), each decision variable can be optimized independently. In rotated problems, on the other hand, improvements require the simultaneous modification of all decision variables which are conditionally dependent on one another. Given the prevalence of conditional dependencies in real-world applications, it is interesting to note that there exist relatively few rotationally invariant operators in active use (Hadka and Reed, 2012a).

2.6 Test Algorithms

In this dissertation, we compare a number of MOEAs designed for many-objective optimization. The following briefly describes the key characteristics of each MOEA, and provides references for additional information.

NSGA-II and SPEA2 (Baselines) The popular NSGA-II (Deb et al., 2000) and SPEA2 (Zitzler et al., 2002a) are two of the oldest MOEAs still in active use today (Coello Coello et al., 2007). NSGA-II is the classical example of a Pareto-based MOEA. SPEA2 differs in its use of Pareto dominance information — the *strength* or fitness of a solution is the number of competing solutions it dominates. Given their sustained popularity in the literature, they are included as baseline algorithms from which to compare more recent contributions.

ϵ-MOEA (Pareto Front Approximation) ϵ-MOEA (Deb et al., 2002b) was the first MOEA to use the ϵ-dominance relation of Laumanns et al. (2002) to provide guarantees of convergence and diversity. It is the only steady-state MOEA tested. The term *steady state* describes EAs and MOEAs that only replace one solution in the population during each iteration of the algorithm. This is in contrast to *generational* algorithms, which replace the entire population in a single iteration. Note also that the Borg MOEA (see Chapter 3) draws on ϵ-MOEA's highly efficient algorithmic structure in its implementation.

ϵ-NSGA-II (Pareto Front Approximation) ϵ-NSGA-II (Kollat and Reed, 2006) is another popular MOEA that combines NSGA-II, an ϵ-dominance archive, adaptive population sizing, and time continuation (Goldberg, 1989b; Srivastava, 2002). In general, MOEAs use a fixed population size and assume the user has specified a population size appropriate for the given problem. ϵ-NSGA-II attempts to adapt the population size relative to the problem difficulty. In addition, *time continuation* is used to trigger a series of connected runs in an attempt to improve search quality. Adaptive population sizing and time continuation are discussed in more detail in Chapter 3. ϵ-NSGA-II has been applied successfully to a broad array of real-world many-objective problems (Kollat and Reed, 2006, 2007; Kasprzyk et al., 2009; Ferringer et al., 2009; Kasprzyk et al., 2011; Kollat et al., 2011).

MOEA/D (Aggregate Fitness Functions) MOEA/D (Zhang et al., 2009a) is a recently-introduced MOEA that uses aggregate functions, but attempts to avoid the pitfalls in prior aggregation approaches (Coello Coello et al., 2007; Wagner et al., 2007) by simultaneously solving many single-objective Chebyshev decompositions of many-objective problems in a single run. Since its introduction, MOEA/D has established itself as a benchmark for new MOEAs by winning the 2009 IEEE Congress on Evolutionary Computation (CEC 2009) competition (Zhang and Suganthan, 2009).

IBEA (Indicator-Based Method) Indicator-based methods work by replacing the Pareto dominance relation with the indicator function. IBEA (Zitzler and Künzli, 2004) uses the hypervolume measure, which avoids active diversity maintenance by not using an explicitly diversity preserving mechanism.

GDE3 (Rotationally Invariant) GDE3 (Kukkonen and Lampinen, 2005) is a multi-objective variant of differential evolution (DE). GDE3 (and DE in general) is notable for *rotationally invariant* operators — they produce offspring independent of the orientation of

the fitness landscape — which is important for problems with high degrees of conditional dependence among its decision variables (Iorio and Li, 2008). GDE3 was a strong competitor in the CEC 2009 competition (Zhang and Suganthan, 2009).

OMOPSO (Pareto Front Approximation) OMOPSO (Sierra and Coello Coello, 2005) is one of the most successful multiobjective particle swarm optimization (PSO) algorithms to date. It is notable for being the first multiobjective PSO algorithm to include ϵ-dominance as a means to solve many-objective problems. OMOPSO thus provides a representative baseline from the PSO class of algorithms.

2.7 Test Problems

The 33 instances of 18 unconstrained, real-valued multiobjective test problems listed in Table 2.1 are used in this dissertation to test the MOEAs. Also shown are the ϵ values used for ϵ-dominance. The UF1-UF13 problems are the unconstrained problems used during the IEEE Congress on Evolutionary Computation (CEC) competition held in 2009 (Zhang et al., 2009b). UF11 and UF12 are rotated instances of the 5D DTLZ2 and DTLZ3 test problems, respectively (Deb et al., 2001). UF13 is the 5D WFG1 test problem (Huband et al., 2006). Appendix A shows example reference sets for these problems. The DTLZ problems are from a set of scalable test problems (Deb et al., 2001). In this dissertation, these problems are tested with 2, 4, 6 and 8 objectives. Table 2.1 also lists the ϵ values used for ϵ-dominance. For the scalable DTLZ test problems, the ϵ values used were 0.01, 0.15, 0.25 and 0.35 for 2, 4, 6 and 8 objectives, respectively.

The conference version of the DTLZ suite (Deb et al., 2002b) omits two problems and relabels another. This dissertation along with most other studies use the problems and names defined in Deb et al. (2001). DTLZ5 and DTLZ6 were omitted since the original problem definitions produce Pareto fronts differing from the published analytical solutions with four or more objectives. This issue was identified by Huband et al. (2006) and corrected in Deb and Saxena (2006) by including additional problem constraints. DTLZ8 and DTLZ9 also include side constraints and were consequently omitted from this dissertation.

2.8 Measuring Quality

When running MOEAs on a MOP, the MOEA outputs an approximation of the Pareto optimal set and Pareto front. The approximation of the Pareto front, called the *approximation set*, can be used to measure the quality of an MOEA on a particular problem. In some situations, such as with contrived test problems, a *reference set* of the globally optimal solutions may be known. If known, the reference set can be used to measure the absolute performance of an MOEA. If not known, the approximation sets from multiple MOEAs can be compared to determine their relative quality.

Table 2.1: The test problems used throughout this dissertation along with key properties.

Problem	M	L	Properties	ϵ
UF1	2	30	Complicated Pareto Set	0.001
UF2	2	30	Complicated Pareto Set	0.005
UF3	2	30	Complicated Pareto Set	0.0008
UF4	2	30	Complicated Pareto Set	0.005
UF5	2	30	Complicated Pareto Set, Discontinuous	0.000001
UF6	2	30	Complicated Pareto Set, Discontinuous	0.000001
UF7	2	30	Complicated Pareto Set	0.005
UF8	3	30	Complicated Pareto Set	0.0045
UF9	3	30	Complicated Pareto Set, Discontinuous	0.008
UF10	3	30	Complicated Pareto Set	0.001
UF11	5	30	DTLZ2 5D Rotated	0.2
UF12	5	30	DTLZ3 5D Rotated	0.2
UF13	5	30	WFG1 5D	0.2
DTLZ1	2-8	M+4	Multimodal, Separable	0.01-0.35
DTLZ2	2-8	M+9	Concave, Separable	0.01-0.35
DTLZ3	2-8	M+9	Multimodal, Concave, Separable	0.01-0.35
DTLZ4	2-8	M+9	Concave, Separable	0.01-0.35
DTLZ7	2-8	M+19	Discontinuous, Separable	0.01-0.35

There is no consensus in the literature of the appropriate procedure with which to compare approximation sets. These procedures, called *performance metrics*, come in two forms: (1) unary and (2) binary performance metrics (Zitzler et al., 2002c). Unary performance metrics produce a single numeric value with which to compare approximation sets. Unary performance metrics have the advantage of permitting the comparison of approximation sets without requiring the actual approximation set, as one need only compare the numeric values. Binary performance metrics, on the other hand, compare pairs of approximation sets, identifying which of the two approximation sets is superior. In order to allow comparisons across studies, this dissertation uses only unary performance metrics.

Zitzler et al. (2002b) contend that the number of unary performance metrics required to determine if one approximation set is preferred over another must be at least the number of objectives in the problem. Because different MOEAs tend to perform better in different metrics (Bosman and Thierens, 2003), Deb and Jain (2002) suggest only using metrics for the two main functional objectives of MOEAs: proximity and diversity. The following outlines several of the commonly-used unary performance metrics. For details of these performance metrics see Coello Coello et al. (2007).

Hypervolume As shown in Figure 2.7, the hypervolume metric computes the volume of the space dominated by the approximation set. This volume is bounded by a reference point, which is usually set by finding the nadir point (i.e., the worst-case objective value for each

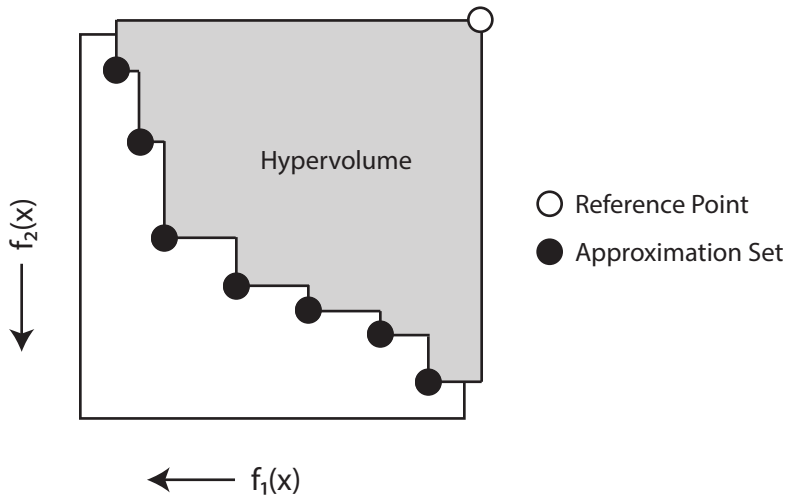


Figure 2.7: Hypervolume measures the volume of the space dominated by the approximation set, bounded by a reference point. This reference point is typically the nadir point (i.e., the worst-case value for each objective) of the reference set plus some fixed delta. This delta ensures extremal points contribute non-zero hypervolume.

objective) of the reference set plus some fixed increment. This fixed increment is necessary to allow the extremal points in the approximation set to contribute to the hypervolume. Knowles and Corne (2002) suggest the hypervolume metric because it is compatible with the outperformance relations, scale independent, intuitive, and can reflect the degree of outperformance between two approximation sets.

The major disadvantage of the hypervolume metric is its runtime complexity of $O(n^{M-1})$, where n is the size of the non-dominated set. However, Beume and Rudolph (2006) provide an implementation with runtime $O(n \log n + n^{M/2})$ based on the Klee's measure algorithm by Overmars and Yap. This implementation permits computing the hypervolume metric on moderately sized non-dominated sets up to $M = 8$ objectives in a reasonable amount of time. Further improvements by While et al. (2012) improve the expected runtime further, allowing the efficient calculation of hypervolume with ten or more objectives.

Generational Distance Generational distance (GD) is the average distance from every solution in the approximation set to the nearest solution in the reference set, as shown in Figure 2.8. As such, it measures proximity to the reference set. GD by itself can be misleading, as an approximation set containing a single solution in close proximity to the reference set produces low GD measurements, and is often combined with diversity measures in practice (Hadka and Reed, 2012b).

Inverted Generational Distance As its name indicates, the inverted generational distance (IGD) is the inverse of GD — it is the average distance from every solution in the reference set to the nearest solution in the approximation set. IGD measures diversity, as

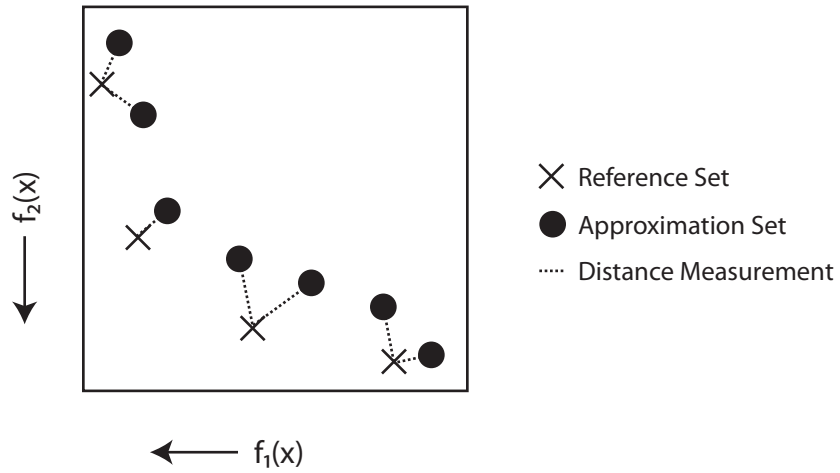


Figure 2.8: Generational distance is the average distance from every solution in the approximation set to the nearest solution in the reference set.

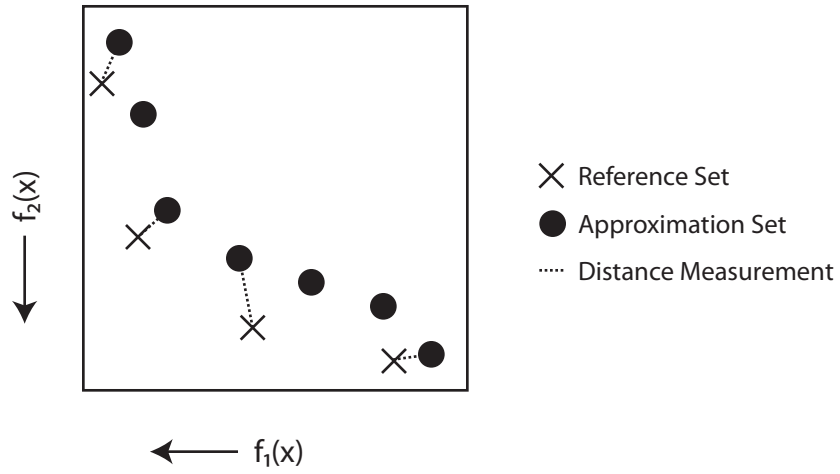


Figure 2.9: Inverted generational distance is the average distance from every solution in the reference set to the nearest solution in the approximation set.

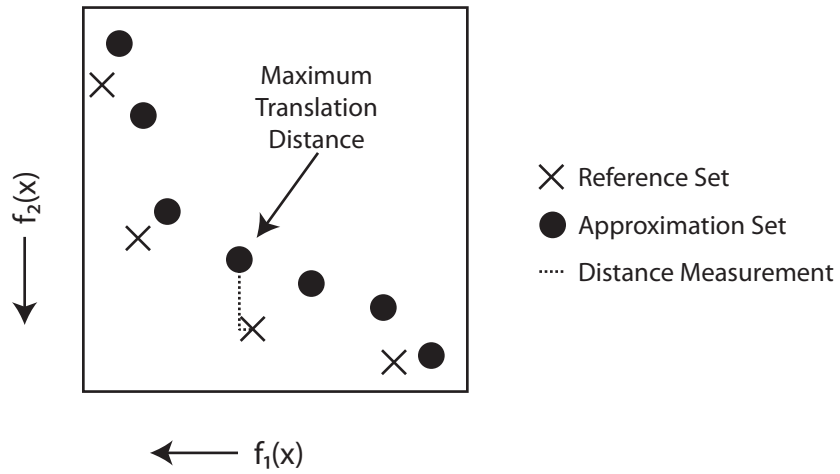


Figure 2.10: ϵ_+-indicator (also known as the additive ϵ-indicator) is the smallest distance ϵ that the approximation set must be translated by in order to completely dominate the reference set (Coello Coello et al., 2007).

shown in Figure 2.9, since an approximation set is required to have solutions near each reference set point in order to achieve low IGD measurements (Coello Coello et al., 2007).

ϵ_+-Indicator The additive ϵ-indicator (ϵ_+-indicator) measures the smallest distance ϵ that the approximation set must be translated by in order to completely dominate the reference set, as shown in Figure 2.10. One observes that good proximity and good diversity both result in low ϵ values, as the distance that the approximation needs to be translated is reduced. However, if there is a region of the reference set that is poorly approximated by the solutions in the approximation set, a large ϵ is required. Therefore, we claim the ϵ_+-indicator measures the *consistency* of an approximation set (Hadka and Reed, 2012a). An approximation set must be free from large gaps or regions of poor approximation in order to be consistent.

Spacing Spacing, shown in Figure 2.11, measures the uniformity of the spacing between solutions in an approximation set (Coello Coello et al., 2007). An approximation set that is well-spaced will not contain dense clusters of solutions separated by large empty expanses. Note that, since spacing does not involve a reference set in its calculation, an approximation can register good spacing while having poor proximity to the reference set. It is therefore recommended to use spacing in conjunction with a performance metric for proximity.

In this dissertation, we have chosen to present results only for GD, hypervolume and ϵ_+-indicator. These three metrics record proximity, diversity and consistency, respectively, which we claim are the three main functional objectives of MOEAs (Fonseca and Fleming, 1996). Figure 2.12 provides a graphical representation of the importance of the ϵ_+-indicator

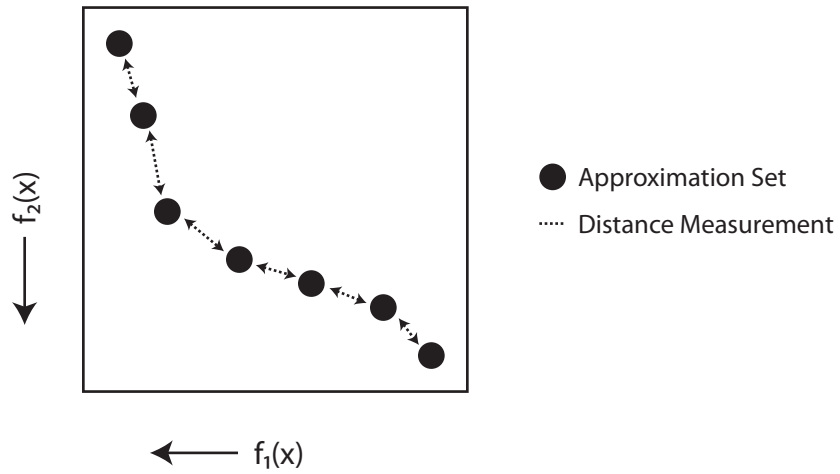


Figure 2.11: Spacing measures the uniformity of the spacing between solutions in an approximation set.

and consistency. MOEAs are expected to produce high-quality solutions covering the entire extent of the tradeoff surface, with few gaps or regions of poor approximation.

In order to report these performance metrics consistently, all performance metrics are normalized. This normalization converts all performance metrics to reside in the range $[0, 1]$, with 1 representing the optimal value. First, the reference set is normalized by its minimum and maximum bounds so that all points in the reference set lie in $[0, 1]^N$, the N-dimensional unit hypercube. Second, each approximation set is normalized using the same bounds. Third, the performance metrics are calculated using these normalized sets. Finally, the performance metrics are transformed by the following equations to ensure a value of 1 represents the optimal value achievable by the metric. Hypervolume is transformed with:

$$\mathcal{M}(A_p^s) = \widehat{\mathcal{M}}(A_p^s)/\mathcal{M}^*, \tag{2.5}$$

where $\widehat{\mathcal{M}}$ represents the raw metric value. GD and the ϵ_+-indicator are transformed with:

$$\mathcal{M}(A_p^s) = \max(1 - \widehat{\mathcal{M}}(A_p^s), 0). \tag{2.6}$$

When solving test problems, such as those listed in Table 2.1, the reference set is known analytically. For most real-world problems, however, the reference set is not available. In these situations, it is often necessary to construct a reference set from the union of all approximation sets generated during experimentation. Then, performance metrics can be evaluated relative to this combined reference set.

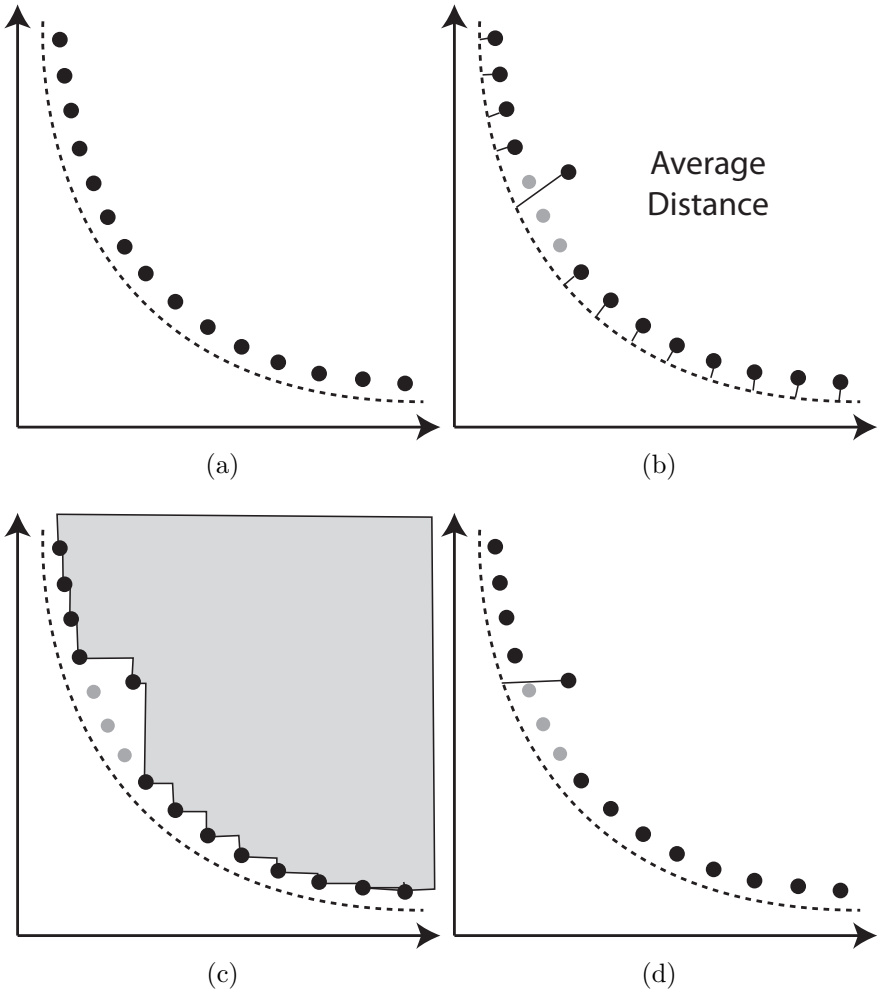


Figure 2.12: Demonstrates the importance of ϵ-indicator as a measure of consistency. (a) A good approximation set to the reference set, indicated by the dashed line. (b) Generational distance averages the distance between the approximation set and reference set, reducing the impact of large gaps. The missing points are shaded light gray. (c) The change in hypervolume due to a gap is small relative to the entire hypervolume. (d) ϵ-Indicator easily identifies the gap, reporting a metric 2-3 times worse in this example.

Chapter 3

Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework

This chapter is drawn from the following paper: "Hadka, D. and Reed, P. (2012). *Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework.* Evolutionary Computation. In-Press."

This chapter introduces the Borg MOEA for many-objective optimization. The Borg MOEA combines ϵ-dominance, a measure of convergence speed named ϵ-progress, randomized restarts, and auto-adaptive multioperator recombination into a unified optimization framework. A comparative study on 33 instances of 18 test problems from the DTLZ, WFG, and CEC 2009 test suites demonstrates that the Borg MOEA meets or exceeds 6 state-of-the-art MOEAs on the majority of the tested problems. Performance for each test problem is evaluated using a 1000 point Latin hypercube sampling of each of the algorithm's feasible parameterization space. The statistical performance of every sampled MOEA parameterization is evaluated using 50 replicate random seed trials. The Borg MOEA is not a single algorithm; instead, it represents a class of algorithms whose operators are adaptively selected based on the problem. The adaptive discovery of key operators is of particular importance for benchmarking how variation operators enhance search for complex many-objective problems.

The remainder of this chapter is organized as follows. Section 3.1 presents the inner workings of the Borg MOEA in detail. Section 3.2 analyzes the algorithm's runtime complexity and details the conditions necessary to guarantee convergence. Section 3.3 presents the results of a comparative study between the Borg MOEA and the 6 state-of-the-art MOEAs listed in Section 2.6.

3.1 The Borg MOEA

The Borg MOEA is designed specifically for handling complex many-objective problems where our primary future focus will be on advancing severely challenging real-world appli-

cations. In order to facilitate these design goals, the Borg MOEA assimilates several design principles from existing MOEAs and introduces several novel components. These components include:

1. an ϵ-box dominance archive for maintaining convergence and diversity throughout search;
2. ϵ-progress, which is a computationally efficient measure of search progression and stagnation introduced in this study;
3. an adaptive population sizing operator based on ϵ-NSGA-II's (Kollat and Reed, 2006) use of time continuation to maintain search diversity and to facilitate escape from local optima;
4. multiple recombination operators to enhance search across a wide assortment of problem domains; and
5. the steady-state, elitist model of ϵ-MOEA (Deb et al., 2003), which can be easily extended for use on parallel architectures.

Each of these components is discussed individually in Sections 3.1.1-3.1.4. Section 3.1.5 discusses how these individual components are combined to form the Borg MOEA.

3.1.1 ϵ-Dominance Archive

As discussed in Section 2.5, deterioration is a fundamental issue encountered by MOEAs. The dominance resistance encountered in many-objective optimization only serves to exacerbate deterioration. Rudolph (1998) and Rudolph and Agapie (2000) presented a selection strategy for a fixed-size archive that avoids deterioration. However, Laumanns et al. (2002) noted that while their selection strategy guarantees convergence to the true Pareto-optimal front, their approach was unable to guarantee a diverse set of Pareto-optimal solutions. As a result of these observations, Laumanns et al. (2002) developed the ϵ-dominance archive in order to guarantee simultaneous convergence and diversity in MOEAs.

Definition 4. *For a given $\epsilon > 0$, a vector $\mathbf{u} = (u_1, u_2, \ldots, u_M)$ **ϵ-dominates** another vector $\mathbf{v} = (v_1, v_2, \ldots, v_M)$ if and only if $\forall i \in \{1, 2, \ldots, M\}, u_i \leq v_i + \epsilon$ and $\exists j \in \{1, 2, \ldots, M\}, u_j < v_j + \epsilon$.*

In addition to the theoretical benefits of guaranteed convergence and diversity, ϵ-dominance provides a minimum resolution which effectively bounds the archive size. This is of practical importance to decision-makers, who are able to define ϵ using domain-specific knowledge of their precision goals or computational limits (Kollat and Reed, 2007; Kasprzyk et al., 2009). In practice, it is useful to specify different ϵ values for each objective; however, without loss of generality, we use a single ϵ value to improve the clarity of this study.

A variant called the *ϵ-box dominance archive* is used in the ϵ-MOEA and ϵ-NSGA-II algorithms by Deb et al. (2003) and Kollat and Reed (2007), respectively. The ϵ-box dominance

Algorithm 1: ϵ-Box Dominance Archive Update Method

Input: The new solution $\mathbf{x}$ being added to the archive.
Output: true if $\mathbf{x}$ is added to the archive; false otherwise.

```
1 foreach solution y in the archive do
2     if x ≺_ε y then
3         remove y from the archive;
4     else if y ≺_ε x then
5         return false;
6 add x to the archive;
7 return true;
```

relation is defined below and the archive update procedure is outlined in Algorithm 1. The archive update procedure is executed once for every solution generated by the MOEA.

Definition 5. *For a given $\epsilon > 0$, a vector $\mathbf{u} = (u_1, u_2, \ldots, u_M)$ **ϵ-box dominates** another vector $\mathbf{v} = (v_1, v_2, \ldots, v_M)$ if and only if one of the following occurs*

1. $\left\lfloor \frac{\mathbf{u}}{\epsilon} \right\rfloor \prec \left\lfloor \frac{\mathbf{v}}{\epsilon} \right\rfloor$, *or*
2. $\left\lfloor \frac{\mathbf{u}}{\epsilon} \right\rfloor = \left\lfloor \frac{\mathbf{v}}{\epsilon} \right\rfloor$ *and* $\left\| \mathbf{u} - \epsilon \left\lfloor \frac{\mathbf{u}}{\epsilon} \right\rfloor \right\| < \left\| \mathbf{v} - \epsilon \left\lfloor \frac{\mathbf{v}}{\epsilon} \right\rfloor \right\|$.

This is denoted by $\mathbf{u} \prec_\epsilon \mathbf{v}$.

Conceptually, the ϵ-box dominance archive divides the objective space into hyperboxes with side-length ϵ, called *ϵ-boxes*. The ϵ-box in which a solution resides is determined using the *ϵ-box index vector*. We use the notation $\left\lfloor \frac{\mathbf{u}}{\epsilon} \right\rfloor = \left(\left\lfloor \frac{u_1}{\epsilon} \right\rfloor, \left\lfloor \frac{u_2}{\epsilon} \right\rfloor, \ldots, \left\lfloor \frac{u_M}{\epsilon} \right\rfloor \right)$ for computing the ϵ-box index vector, where $\lfloor \cdot \rfloor$ is the floor function. As seen in Definition 5, dominance is determined using this index vector rather than the objective values. Case 2 in Definition 5 covers the situation in which two or more solutions reside in the same ϵ-box. In this situation, the solution nearest the optimal corner (i.e., bottom-left corner if minimized) of the ϵ-box dominates any other solutions in the same ϵ-box.

3.1.2 ϵ-Progress

While the ϵ-box dominance archive guarantees convergence and diversity, this guarantee is subject to the solutions produced by the MOEA. MOEAs tend to fail on multimodal problems due to preconvergence to local optima causing search to stagnate. In this section, we introduce a computationally efficient extension to the ϵ-box dominance archive for measuring search progression called *ϵ-progress*. Consequently, the inability of an MOEA to maintain ϵ-progress indicates search stagnation, which can subsequently trigger routines for reviving search.

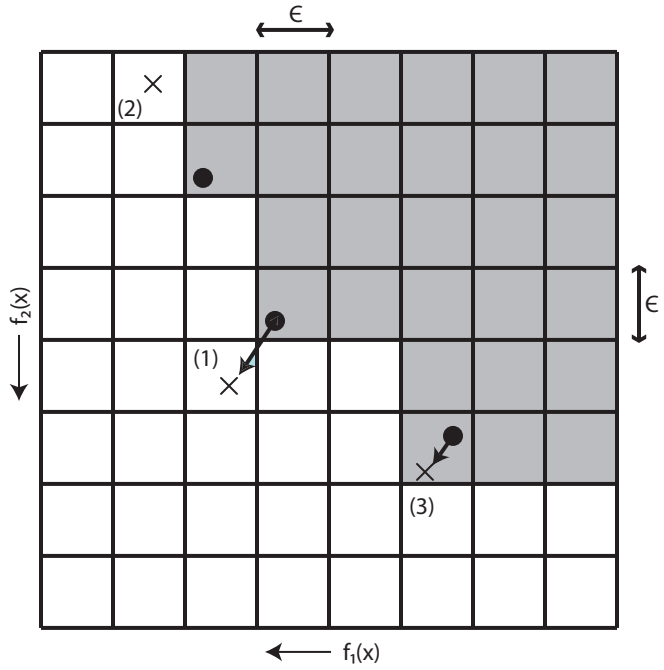


Figure 3.1: 2D example depicting how ϵ-progress is measured. Existing archive members are indicated by $\bullet$, and the ϵ-boxes dominated by these members are shaded gray. New solutions being added to the archive are indicated by $\times$. Cases (1) and (2) depict occurrences of ϵ-progress. The new solutions reside in previously unoccupied ϵ-boxes. Case (3) shows the situation in which the new solution is accepted into the archive, but since it resides in an occupied ϵ-box it does not count towards ϵ-progress — the improvement is below the threshold ϵ.

Definition 6. ***ϵ-Progress*** *occurs when a solution* $\mathbf{x}$ *passed to the update procedure outlined in Algorithm 1 is accepted into the archive such that no existing member of the archive existed with the same ϵ-box index vector.*

ϵ-Progress supplements the use of ϵ as the problem resolution by mandating ϵ as the minimum threshold for improvement. An MOEA must periodically produce at least one solution whose improvement exceeds this threshold to avoid stagnation. If stagnation is detected, appropriate action can be taken to either revive search or terminate the algorithm.

Figure 3.1 demonstrates ϵ-progress on a 2D example. Existing archive members are indicated by $\bullet$, and the ϵ-boxes dominated by these members are shaded gray. New solutions being added to the archive are indicated by $\times$. Cases (1) and (2) depict occurrences of ϵ-progress. The new solutions reside in previously unoccupied ϵ-boxes. Case (3) shows the situation in which the new solution is accepted into the archive, but since it resides in an occupied ϵ-box it does not count towards ϵ-progress — the improvement is below the threshold ϵ.

Extending the ϵ-box dominance archive in Algorithm 1 to include ϵ-progress is straightforward. In this study, the ϵ-box dominance archive increments a counter every time ϵ-progress occurs. This counter is periodically checked after a user-specified number of evaluations. If the counter is unchanged from the previous check, then the MOEA failed to produce significant improvements, and the restart mechanism discussed in Section 3.1.3 is triggered.

3.1.3 Restarts

Restarts are a mechanism for reviving search after stagnation is detected using ϵ-progress. In the Borg MOEA, a restart consists of three actions:

1. the search population size is adapted to remain proportional to the archive size;
2. the tournament selection size is adapted to maintain elitist selection; and
3. the population is emptied and repopulated with solutions from the archive, with any remaining slots filled by mutated archive solutions.

Each of these three functions utilized in restarts are described in more detail below.

Adaptive Population Sizing Tang et al. (2006) observed that maintaining a population size proportional to the archive size helped escape local optima on a highly multi-modal real-world problem. This mechanism of adapting the population size is built into the ϵ-NSGA-II algorithm by Kollat and Reed (2006) via the use of the population-to-archive ratio γ (ϵ-NSGA-II literature refers to this ratio as the *injection rate*). The population-to-archive ratio specifies the ratio of the population size to the archive size:

$$\gamma = \frac{\text{population size}}{\text{archive size}} \geq 1. \tag{3.1}$$

The Borg MOEA utilizes the same adaptive population sizing strategy as ϵ-NSGA-II, except that the population-to-archive ratio is maintained throughout the run. At any point during the execution of the algorithm, if the population-to-archive ratio differs from γ by more than 25%, the population size is adapted. Figure 3.2 outlines the logic of triggering restarts by ϵ-progress and the population-to-archive ratio.

This strategy ensures the population size remains commensurate with the Pareto front discovered by the MOEA. By using the archive size as a proxy for problem difficulty, we assume the population should grow proportionally with problem difficulty based on the theoretical recommendations of Horn (1995) and Mahfoud (1995).

Adaptive Tournament Size The Borg MOEA is designed such that it maintains tournament sizes to be τ, a fixed percentage of the population size, after every restart:

$$\text{tournament size} = \max\left(2, \lfloor \tau(\gamma A) \rfloor\right), \tag{3.2}$$

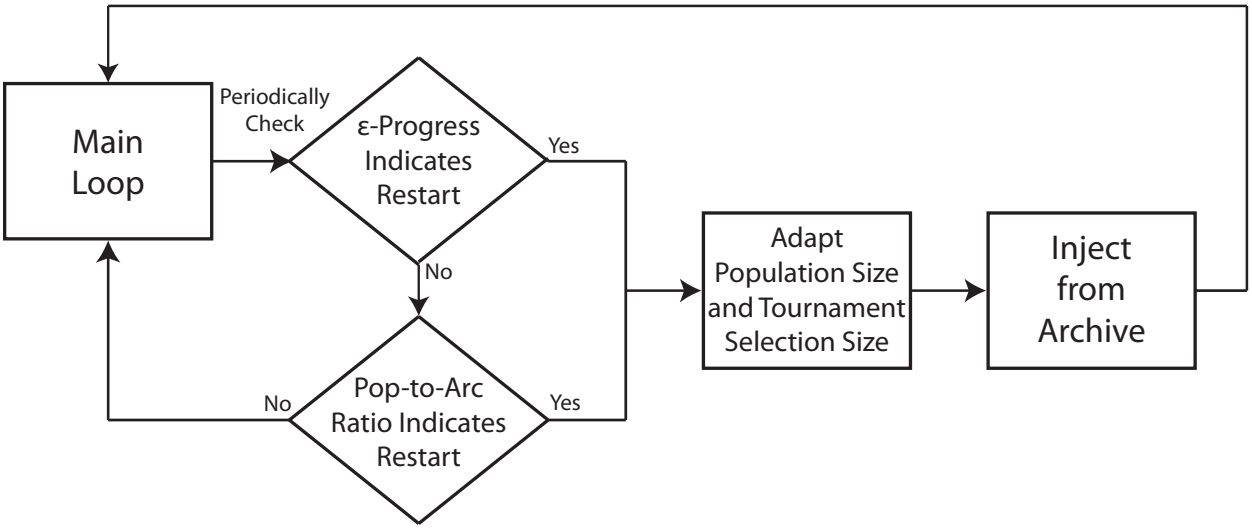


Figure 3.2: Flowchart of the Borg MOEA's restart logic. After a certain number of evaluations, the MOEA breaks out of its main loop to check if ϵ-progress or the population-to-archive ratio indicate a restart is required. If a restart is required, the population is resized and filled with all members of the archive. Any remaining population slots are filled with solutions selected randomly from the archive and mutated using uniform mutation applied with probability $1/L$. In addition, the tournament selection size is adjusted to account for the new population size. Finally, the MOEA's main loop is resumed.

where A is the size of the archive. As Deb (2001) discusses, the concept of selection pressure is important in understanding the convergence behavior of EAs, but its formulation is not readily applicable to multiobjective optimization. Whereas selection pressure originally measured the probability of selecting the i-th best individual from a population (Bäck, 1994), the multiobjective equivalent can be formulated as the probability of selecting a solution from the i-th best rank. If we assume that the proportion of non-dominated solutions in the population is approximately $1/\gamma$ after a restart, then the probability of binary tournament selection choosing a non-dominated member when $\gamma = 4$ is $1-(1-1/\gamma)^2 = 1-(\frac{3}{4})^2 = 0.44$. If instead $\gamma = 8$, then this probability decreases to $1-(\frac{7}{8})^2 = 0.23$, or roughly half as before. In order to maintain the same multiobjective selection pressure, the tournament size must be increased to 4, resulting in a selection probability of $1-(\frac{7}{8})^4 = 0.41$. In this manner, τ governs the tournament size as the population dynamics increase the population size beyond the initial minimum value. Note that $\tau = 0$ can be used to enforce binary tournaments regardless of the population size.

Injection The idea of injection is derived from the work of Goldberg (1989b) and Srivastava (2002) who exploit *time continuation*. Time continuation uses multiple-epoch runs instead of the single-epoch run typically employed by MOEAs. Multiple-epoch runs are characterized by periodically emptying the population, retaining the best solution(s), and repopulating with new randomly-generated solutions. For multiobjective problems, Kollat and Reed (2006) introduced *injection*, which involves refilling the population with all members of the archive. Any remaining slots in the population are filled with new randomly-generated solutions.

After some experimentation on the DTLZ (Deb et al., 2001), WFG (Huband et al., 2006), and CEC 2009 (Zhang et al., 2009b) test suites, we observed that filling the remaining slots with solutions selected randomly from the archive and mutated using uniform mutation applied with probability $1/L$ achieved significantly better results. This is supported by the work of Schaffer et al. (1989) and others showing the dependence of effective mutation rates upon the number of decision variables L.

Figure 3.3 illustrates how a population evolves throughout the execution of the Borg MOEA as a result of the restart mechanism. Pseudocode for the restart mechanism is presented in Algorithm 2.

3.1.4 Auto-Adaptive Multi-Operator Recombination

One of the problems encountered when using MOEAs in real-world contexts is the inability to know *a priori* which recombination operator performs best on a given problem. Vrugt and Robinson (2007) and Vrugt et al. (2009) address this issue by introducing an adaptive multi-operator hybrid called AMALGAM. The adaptability and reliability of AMALGAM was demonstrated on 10 multiobjective test problems in Vrugt and Robinson (2007) and a complex hydrologic model calibration problem in Zhang et al. (2010).

The idea is to establish a feedback loop in which operators that produce more successful offspring are rewarded by increasing the number of offspring produced by that operator.

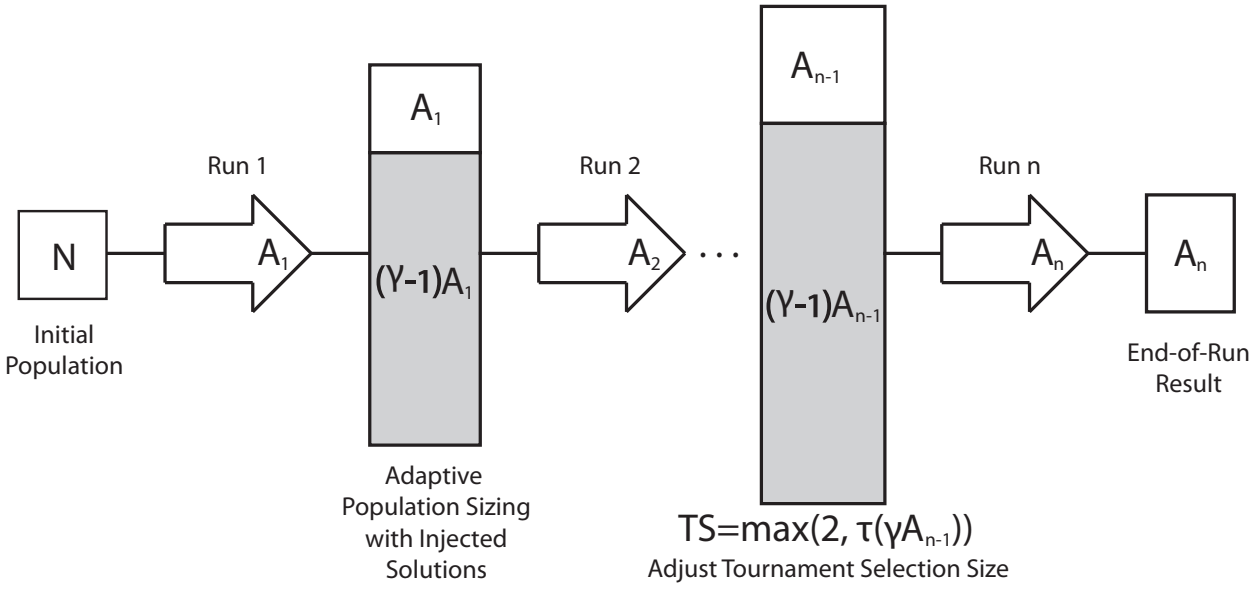


Figure 3.3: Illustration of how a population evolves from multiple restarts, forming what is known as "connected runs." With an initial population of size N, the MOEA is run until the first restart is triggered. At this point, the population is emptied and filled with the current archive, A_1. Next, the remaining slots in the resized population, shown in gray, are filled with solutions selected randomly from A_1 and mutated using uniform mutation applied with probability $1/L$. Lastly, the tournament size is adjusted to account for the new population size. This process repeats until termination.

Algorithm 2: Random Restart

Input: The current archive, the population-to-archive ratio γ and the selection ratio τ
Output: The population after random restart

Empty the population;
Fill population with all solutions in the archive;
`// Compute the size of the new population`
new_size $\leftarrow \gamma *$ `size`(archive);
`// Inject mutated archive members into the new population`
while `size`(population) $<$ new_size **do**
    new_solution $\leftarrow$ select randomly one solution from archive;
    Mutate new_solution using uniform mutation applied with probability $1/L$;
    Add new_solution to population;
    Update archive with new_solution;
`// Adjust tournament size to account for the new population size`
Set the tournament size to $\texttt{max}(2, \texttt{floor}(\tau * \text{new\_size}))$;

Given $K > 1$ operators, we maintain the probabilities $\{Q_1, Q_2, \ldots, Q_K\}$, $Q_i \in [0, 1]$, of applying each operator to produce the next offspring. These probabilities are initialized to $Q_i = 1/K$. Periodically, these probabilities are updated by first counting the number of solutions in the ϵ-box dominance archive that were produced by each operator, $\{C_1, C_2, \ldots, C_K\}$, and updating each Q_i by

$$Q_i = \frac{C_i + \varsigma}{\sum_{j=1}^{K}(C_j + \varsigma)}. \tag{3.3}$$

The constant $\varsigma > 0$ prevents the operator probabilities from reaching 0, thus ensuring that no operators are "lost" during the execution of the algorithm. In this study, we use $\varsigma = 1$.

This approach differs from AMALGAM primarily in how the probabilities are updated. Our feedback loop updates the probabilities by counting the number of solutions produced by each operator in the ϵ-box dominance archive. Since AMALGAM is based on NSGA-II, which does not use an archive, it instead counts solutions in the population. This lack of an ϵ-dominance archive makes AMALGAM prone to deterioration on many-objective problems (Laumanns et al., 2002). In addition, since the ϵ-box dominance archive maintains the best solutions in terms of both convergence and diversity, our approach favors operators producing offspring with both of these qualities.

As a result, the Borg MOEA is not a single algorithm but a class of algorithms whose operators are adaptively selected based on the problem and the decision variable encoding. The discovery of key operators is of particular importance to real-world problems where such information is unknown *a priori*. In addition, this is an ideal platform for benchmarking how new variation operators enhance search on complex many-objective problems. Since this study is considering only real-valued test problems, we have selected the following parent-

centric, mean-centric, uniformly distributed and self-adaptive real-valued operators:

- Simulated Binary Crossover (SBX) (Deb and Agrawal, 1994)
- Differential Evolution (DE) (Storn and Price, 1997)
- Parent-Centric Crossover (PCX) (Deb et al., 2002a)
- Unimodal Normal Distribution Crossover (UNDX) (Kita et al., 1999)
- Simplex Crossover (SPX) (Tsutsui et al., 1999)
- Uniform Mutation (UM) applied with probability $1/L$

In addition, offspring produced by SBX, DE, PCX, UNDX and SPX are mutated using Polynomial Mutation (PM) (Deb and Agrawal, 1994). Figure 3.4 provides examples showing the offspring distribution generated by each of these operators. These figures show the tendency of SBX, UM and PM to generate solutions along a single axis, which degrades their efficacy on problems with conditional dependencies among its decision variables. DE, PCX, UNDX and SPX do not exhibit this tendency; one can expect these four operators to perform better on rotated, epistatic problems.

3.1.5 The Algorithm

The Borg MOEA combines the components discussed in the previous sections within the ϵ-MOEA algorithm introduced by Deb et al. (2003). The rationale behind selecting ϵ-MOEA is its highly efficient steady-state model. Selection and replacement in ϵ-MOEA is based solely on the dominance relation and requires no expensive ranking, sorting, or truncation. In addition, the steady-state model will support parallelization in future studies without the need for synchronization between generations.

Figure 3.5 is a flowchart of the Borg MOEA's main loop. First, one of the recombination operators is selected using the adaptive multi-operator procedure described in Section 3.1.4. For a recombination operator requiring k parents, 1 parent is selected uniformly at random from the archive. The remaining $k - 1$ parents are selected from the population using tournament selection. The resulting offspring are evaluated and considered for inclusion in the population and archive.

If the offspring dominates one or more population members, the offspring replaces one of these dominated members randomly. If the offspring is dominated by at least one population member, the offspring is not added to the population. Otherwise, the offspring is non-dominated and replaces a randomly-selected member of the population. Inclusion in the archive is determined with the archive update procedure outlined in Section 3.1.1.

Each iteration of this main loop produces one offspring. After a certain number of iterations of this main loop, ϵ-progress and the population-to-archive ratio are checked as described in Section 3.1.3. If a restart is required, the main loop halts and the restart

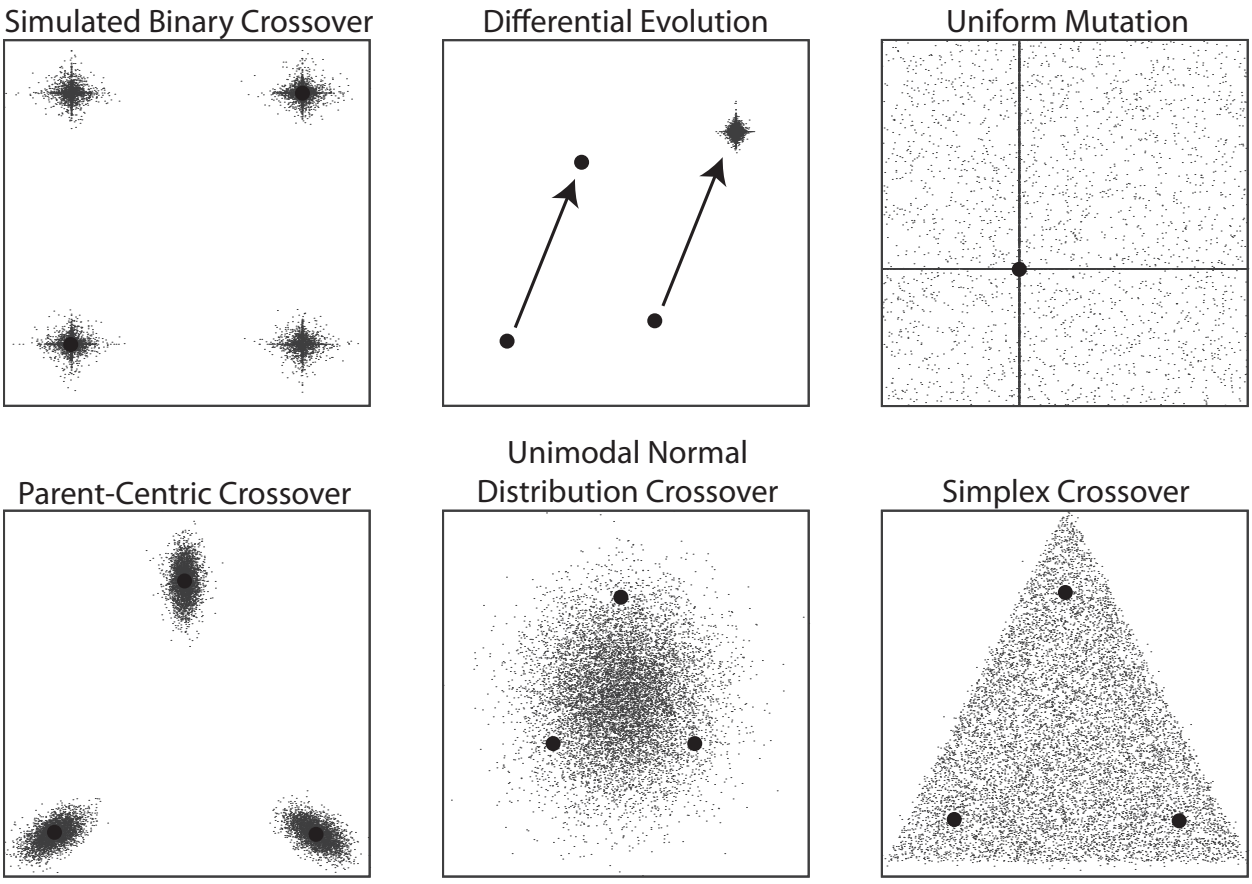


Figure 3.4: Examples showing the offspring distribution of the operators used in this study. Parents are indicated by •. The differential evolution plot depicts the difference vector with arrows.

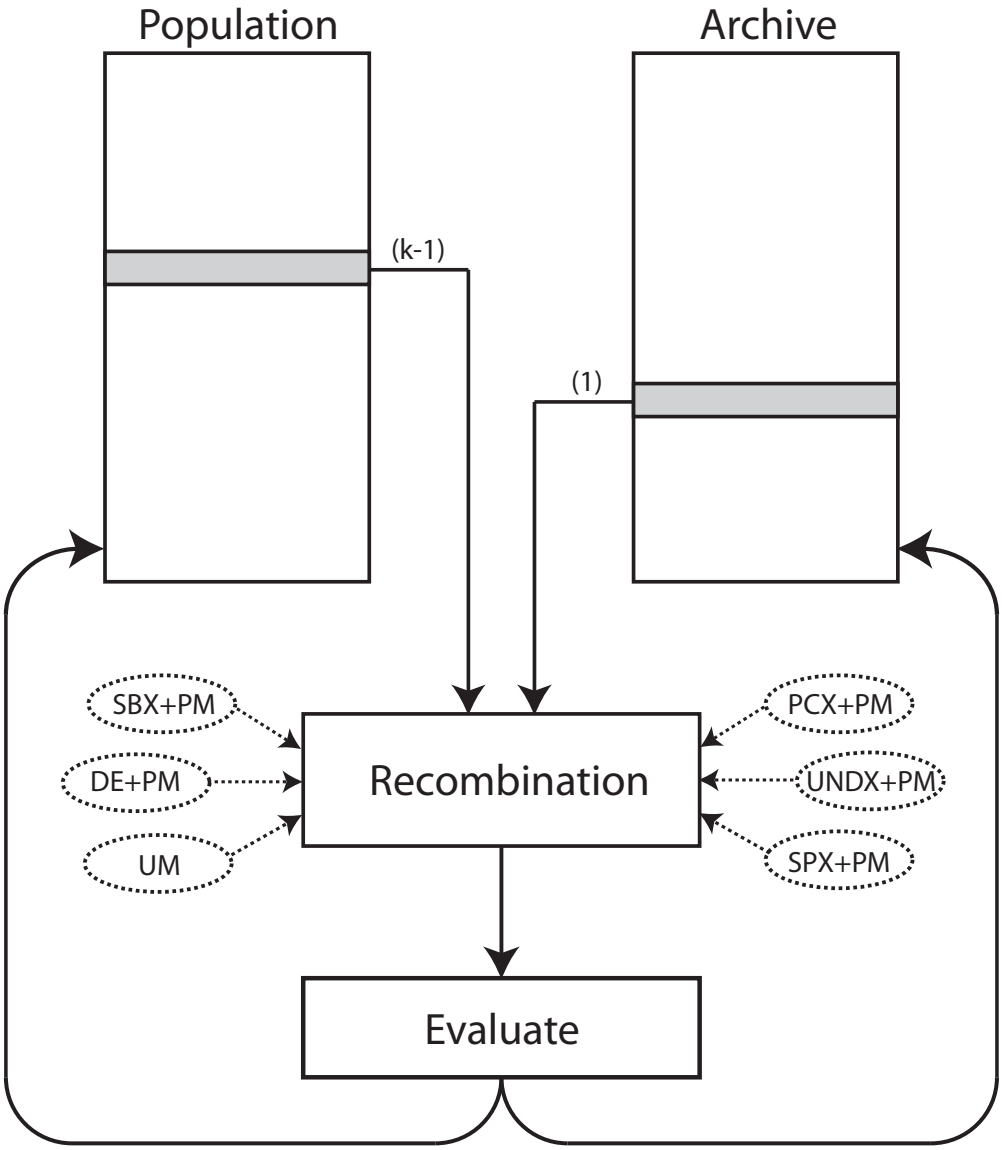


Figure 3.5: Flowchart of the Borg MOEA main loop. First, one of the recombination operators is selected using the adaptive multi-operator procedure described in Section 3.1.4. For a recombination operator requiring k parents, 1 parent is selected uniformly at random from the archive. The remaining $k - 1$ parents are selected from the population using tournament selection. The offspring resulting from this operator are evaluated and then considered for inclusion in the population and archive.

procedure is invoked. Once the restart has completed, the main loop is resumed and this process repeats until termination.

For the comparative analysis in this study, the Borg MOEA terminates after a fixed number of function evaluations. However, in practice, ϵ-progress can be used to terminate the algorithm if no improvements are detected after a specified number of function evaluations.

3.2 Theoretical Characteristics

3.2.1 Runtime Analysis

Consider the runtime computational complexity of the Borg MOEA. For each offspring, dominance checks against the population and archive of sizes P and A, respectively, take time $O(M(P + A))$. However, since the population size is a constant multiple of the archive size, this simplifies to $O(MA)$. For η evaluations, the total runtime of the Borg MOEA is $O(\eta MA)$. Note that we simplified these expressions by assuming selection and recombination take constant time.

Thus, the Borg MOEA is an efficient algorithm that scales linearly with the archive size. Recall from Section 3.1.1 how the archive size is controlled by the value of ϵ. By scaling ϵ, the algorithm can be made to run more efficiently at the cost of producing more approximate representations of the Pareto front. The determination of ϵ is left to the decision-maker, who may use domain-specific knowledge of their significant precision goals or computational limits (Kollat and Reed, 2007).

3.2.2 Proof of Convergence

Exploring the limit behavior of an algorithm as the runtime goes to infinity, $t \to \infty$, is important from a theoretical view. It is not necessary for an algorithm to have guaranteed convergence to be practically useful, but issues like preconvergence and deterioration that arise in many-objective optimization make such results informative. In fact, most MOEAs do not have guaranteed convergence (Laumanns et al., 2002). The main crux of such convergence proofs is the assumption that there exists a non-zero probability of generating Pareto optimal solutions. Using the terminology of Rudolph (1998) and Rudolph and Agapie (2000), the recombination operators must have *diagonal-positive transition matrices*. Since tournament selection operates with replacement and all recombination operators used in this study have a form of mutation in which the entire decision space is reachable, the conditions outlined by Rudolph and Agapie (2000) for diagonal-positive transition matrices are satisfied.

The second necessary condition for guaranteed convergence on a multiobjective problem is *elite preservation* (Rudolph, 1998). As proved by Laumanns et al. (2002), the ϵ-dominance archive satisfies elite preservation. The ϵ-box dominance archive used in this study also satisfies elite preservation using the same logic — a solution in the archive at time t, $\mathbf{x} \in A_t$, is not contained in A_{t+1} if and only if there exists a solution $\mathbf{y} \in A_{t+1}$ with $F(\mathbf{y}) \prec_\epsilon F(\mathbf{x})$ — thus proving the sequence of solutions generated by the Borg MOEA converges completely

and in the mean to the set of minimal elements (the Pareto optimal set) as $t \to \infty$. In addition, Laumanns et al. (2002) proved the ϵ-box dominance archive preserves the diversity of solutions.

3.2.3 Recommended Parameter Values

Appropriate parameterization of the algorithm and operators is important for its efficiency and effectiveness. The following parameterization guidelines are derived from the Latin hypercube sampling performed in Section 3.3 and the suggested operator parameterizations from the literature (refer to the cited papers for the meaning and usage of the parameters).

For the Borg MOEA, it is recommended to use an initial population size of 100, a population-to-archive ratio of $\gamma = 4$ and a selection ratio of $\tau = 0.02$. On the problems tested, the SBX and PM operators performed best with distribution indices less than 100 with SBX applied with probability greater than 0.8. Both PM and UM should be applied with probability $1/L$. DE performed best with a crossover rate and step size of 0.6. For the multiparent operators, Deb et al. (2002a) suggests using 3 parents for PCX and UNDX and $L + 1$ parents for SPX. For PCX, the σ_η and σ_ζ parameters controlling the variance of the resulting distribution should be set to 0.1 (Deb et al., 2002a). For UNDX, use $\sigma_\xi = 0.5$ and $\sigma_\eta = 0.35/\sqrt{L}$ to preserve the mean vector and covariance matrix (Kita et al., 1999). For SPX, the expansion rate should be $\sqrt{P+1}$, where P is the number of parents, to preserve the covariance matrix of the population (Tsutsui et al., 1999).

3.3 Comparative Study

To test the performance of the Borg MOEA, a comparative study between the Borg MOEA, ϵ-MOEA, MOEA/D, GDE3, OMOPSO, IBEA and ϵ-NSGA-II was undertaken using several many-objective test problems from the DTLZ (Deb et al., 2001), WFG (Huband et al., 2006), and CEC 2009 (Zhang et al., 2009b) test problem suites. These top-ranked MOEAs provide a rigorous performance baseline for distinguishing the Borg MOEA's contributions on a set of problems widely accepted in the community for benchmarking performance (Zhang and Suganthan, 2009). Table 2.1 lists the problems explored in this study along with their key properties.

While the figures in this section only show the hypervolume metric, Table 3.1 does include summary results with generational distance and the additive ϵ-indicator (ϵ_+). Generational distance directly measures convergence whereas the ϵ_+-indicator provides a better measure of diversity and consistency (Coello Coello et al., 2007).

Each algorithm was executed 1000 times using parameters produced by a Latin hypercube sampling (LHS) (McKay et al., 1979) across each of the algorithm's feasible parameter range. Each execution of a sampled parameter set was replicated 50 times with different randomly-generated initial populations. The parameters analyzed include the population size, maximum number of objective function evaluations, and the parameters controlling selection and recombination operators. Since certain parameterizations can result in poor

Table 3.1: Statistical comparison of algorithms based on the 75% quantile of the hypervolume, generational distance, and ϵ_+-indicator metrics. +, =, and − indicate Borg's 75% quantile was superior, statistically indifferent from, or inferior to the competing algorithm, respectively.

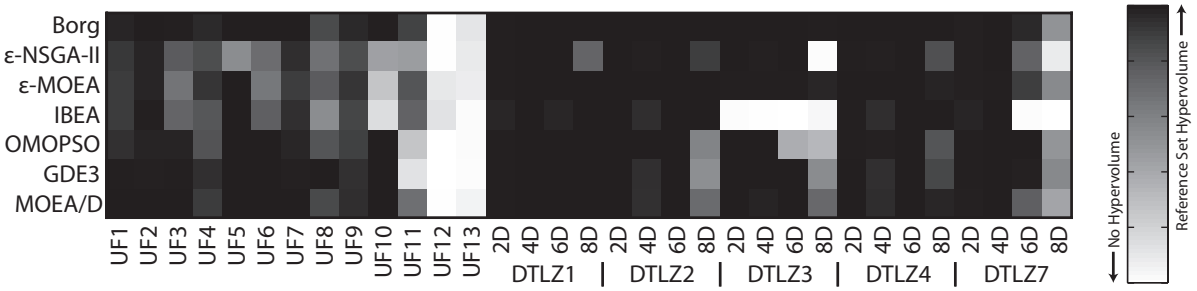
	Hypervolume			Generational Distance			ϵ_+-Indicator		
Algorithm	+	=	−	+	=	−	+	=	−
ϵ-NSGA-II	15	8	10	17	4	12	15	4	14
ϵ-MOEA	16	9	8	24	3	6	17	3	13
IBEA	23	7	3	18	1	14	24	2	7
OMOPSO	24	4	5	25	3	5	22	4	7
GDE3	25	2	6	29	3	1	24	2	7
MOEA/D	25	3	5	27	3	3	24	4	5

performance, the worst performing half of all parameterizations were eliminated from the remainder of this analysis. By analyzing the set of best performing parameters, we measure the performance of an algorithm in terms of solution quality as well as its reliability and controllability across a range of parameterizations.

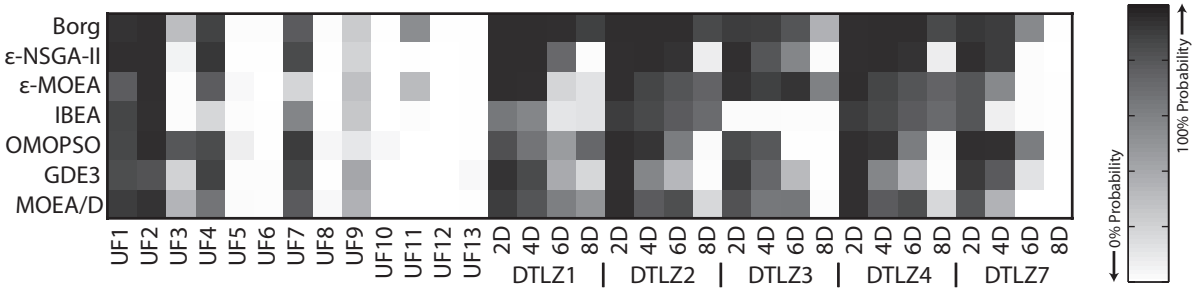
The ranges from which the parameters were sampled is as follows. The number of fitness evaluations was sampled between $[10000, 1000000]$ in order to permit tractable execution times while providing meaningful results. The population size, offspring size, and archive sizes are all sampled between $[10, 1000]$. This range was chosen to encompass the commonly employed "rule-of-thumb" population sizes in MOEA parameterization recommendations. Mutation rate, crossover rate, and step size encompass their entire feasible ranges of $[0, 1]$. Distribution indices for SBX and PM range between $[0, 500]$, which is based on the "sweet spot" identified by Purshouse and Fleming (2007). The ϵ values used by the Borg MOEA, ϵ-MOEA, ϵ-NSGA-II and OMOPSO are shown in Table 2.1.

Table 3.1 provides a summary of the results from this analysis. The Kruskal-Wallis one-way analysis of variance and Mann-Whitney U tests were used to compare the algorithms using the 75% quantile of the hypervolume, generational distance and ϵ_+-indicator metrics with 95% confidence intervals (Sheskin, 2004). These tests help guarantee any difference in the observed value is statistically significant and not a result of random chance. Table 3.1 records the number of problems in which the Borg MOEA outperformed, underperformed or was statistically indifferent from each competing algorithm with respect to the 75% quantile of each metric. The 75% quantile was selected to compare the algorithms at a moderate level of success; however, the results do not differ significantly from the 50% and 90% quantiles. As shown, the Borg MOEA outperformed the competing algorithms on the majority of problem instances, but ϵ-NSGA-II and ϵ-MOEA were strong competitors.

For a more detailed view of the results, we compare the algorithms using their best achieved value and the probability of attaining at least 75% of the reference set hypervolume. The best achieved value, shown in Figure 3.6a, presents the best achieved hypervolume for each algorithm across all seeds and parameters. Figure 3.6b, which shows the probability



(a) Best Achieved



(b) Attainment

Figure 3.6: Best achieved and 75% attainment results from the comparative study. (a) shows the best value achieved by the MOEA across all seeds, where black indicates values near the reference set hypervolume. (b) shows the probability of attaining at least 75% of the reference set hypervolume for each problem. Black indicates 100% probability; white indicates 0% probability.

of attaining at least 75% of the reference set hypervolume, indicates for each algorithm the percentage of its parameters and seeds that reached a moderate level of success (i.e., 75% of the reference set hypervolume). We distinguish between these two measurements since the best achieved value may be a needle-in-the-haystack, where only a small number of parameters or seeds were successful. In this scenario, reporting only the best achieved value hides the fact that the likelihood of producing the best achieved value is low. The attainment measurement distinguishes these cases.

Figure 3.6a shows that across the majority of the tested problem instances, the Borg MOEA is able to produce approximation sets matching or exceeding the quality achieved by the competing algorithms. Only in UF1, UF8, UF12 and DTLZ7 8D is the Borg MOEA slightly outperformed. As GDE3 is the only algorithm outperforming the Borg MOEA on all such cases, this suggests the rotationally-invariant DE operator may prove useful on these instances and consequently an optimal operator choice would be expected to provide some advantage relative to learning. MOEA/D and OMOPSO also show an advantage on the UF1 and 6D DTLZ7, respectively.

Figure 3.6a also shows several algorithms failing on UF12, UF13 and DTLZ3 at higher dimensions. UF12 and UF13 are rotated instances of the 5D DTLZ3 and 5D WFG1 problems. As unrotated DTLZ3 instances cause many MOEAs to fail (Hadka and Reed, 2012b), it is not surprising that UF12 is difficult. What is surprising, however, is that the MOEAs tested in this study with rotationally-invariant operators (e.g., GDE3 and Borg) struggled on UF12, given their good performance on the 6D DTLZ3. In addition, IBEA seems to completely fail on DTLZ3. As IBEA uses SBX and PM, which are the variation operators used by a number of the MOEAs tested in this study, this suggests the hypervolume indicator fails to guide search on this problem. Further investigation of this disparity should be undertaken.

While the majority of the algorithms produce at least one good approximation set on UF3, UF5, UF6, UF8 and UF10, Figure 3.6b shows that the probability of doing so is very low. This demonstrates how reporting only the best attained value may be misleading, as the likelihood of attaining good quality solutions may be extremely low.

Identifying and understanding the root causes of these failures is necessary to improve the reliability of MOEAs. UF5 and UF6 both consist of small, disjoint, finitely sized Pareto sets (Zhang et al., 2009b). These sparse Pareto optimal solutions are separated by large gaps, which appear to cause significant problems for the variation operators, many of which like SBX, PCX and PM favor producing offspring near the parents. It is not immediately obvious which properties of UF3, UF8 and UF10 are causing all tested MOEAs to fail. UF8 and UF10 do share identical Pareto sets and Pareto fronts, which suggests the construction of the Pareto sets and Pareto fronts for these two problems may be the source of such failures.

In summary, the Borg MOEA showed superior performance in both the best attained value and the probability of attaining at least 75% of the reference set hypervolume. This is initial evidence that the Borg MOEA provides superior performance and reliability when compared to other state-of-the-art MOEAs. However, there is still room for improvement on several of the UF test problems for all algorithms, as seen in the attainment results. The difficulties exhibited by UF3, UF5, UF6, UF8 and UF10 should prompt further investigation

and influence the development of additional test problems.

3.3.1 Control Maps

Figures 3.7 and 3.8 provide a more detailed exploration of the algorithms' performance on two specific problem instances, DTLZ2 and DTLZ1, by showing their control maps. These two problem instances are selected since DTLZ2 is one of the easiest problems tested in this study, whereas DTLZ1 is multi-modal and challenging for all of the algorithms. Control maps highlight regions in parameter space whose parameterizations produce approximation sets with hypervolume values near the reference set hypervolume (black regions), and parameterizations that produce poor approximation sets (white regions). In this case, we are plotting population size versus the number of objective function evaluations.

Identifying so-called *"sweet spots"* is of particular interest, which are large regions of high-performing parameterizations (Goldberg, 1998). In Figure 3.7, all algorithms excluding IBEA show reliable parameterization on the 2D DTLZ2 instance. However, as the number of objectives is increased, MOEA/D, GDE3, OMOPSO and IBEA show significant declines in performance. The Borg MOEA, ϵ-MOEA and ϵ-NSGA-II retain a large sweet spot on DTLZ2 instances with up to 8 dimensions, but a small decline in performance is observed on ϵ-MOEA and ϵ-NSGA-II on the 8D DTLZ2 problem. In Figure 3.8, we observe that the Borg MOEA and ϵ-NSGA-II are the only algorithms showing large sweet spots on DTLZ1, even on the 2D instance. The Borg MOEA is the only tested algorithm with a sweet spot on the 8D DTLZ1 instance.

ϵ-MOEA and IBEA have chaotic control maps, with patches of light and dark regions, indicating that specific parameters or parameter combinations are resulting in poor performance. Algorithms whose performance is highly dependent on its parameter selection are expected to be difficult to use on real-world problems, where expensive objective evaluation costs prohibit experimentation to discover correct parameter settings. Utilizing MOEAs with large "sweet spots" is therefore desirable in real-world settings.

For algorithms that do not exhibit large "sweet spots", trends can often be observed to guide better parameter selection. As an example, Figures 3.7 and 3.8 show MOEA/D has a strong dependency on population size. These results suggest that running MOEA/D with larger population sizes will tend to improve its resulting approximation sets. However, since MOEA/D's neighborhood scheme severely increases its runtime as the population size grows, increasing the population size may not be a feasible option. Borg is expected to be insensitive to the initial population size due to its adaptive population sizing scheme. Figures 3.7 and 3.8 confirm this hypothesis. For the Borg MOEA, the number of objective function evaluations is key to improving its performance, suggesting the Borg MOEA will benefit from parallelization. The study of parameterization trends and their impact on controllability is discussed in detail in Chapter 4.

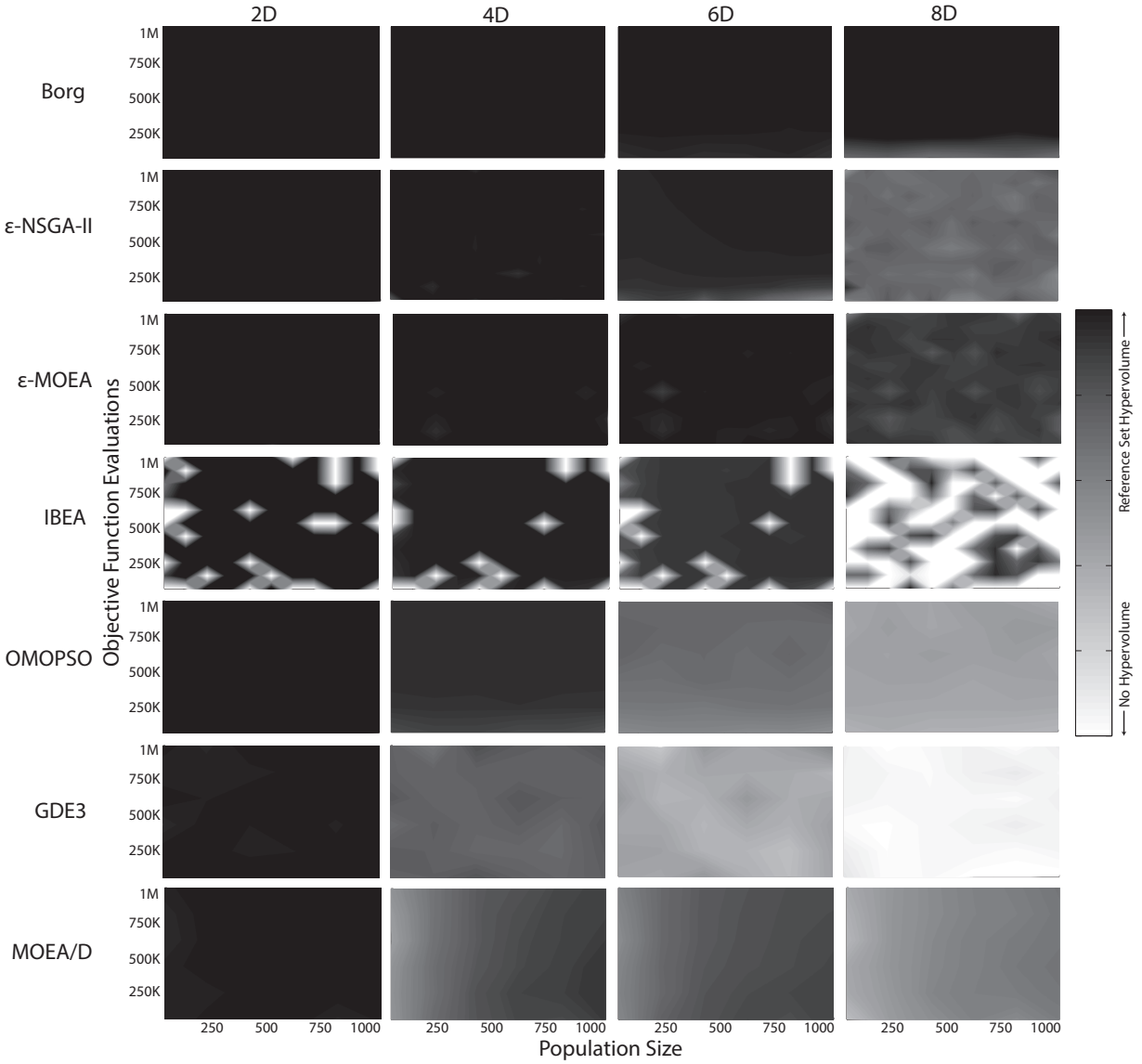


Figure 3.7: Control map showing the relation between population size and number of objective function evaluations on the DTLZ2 problem from 2 to 8 objectives.

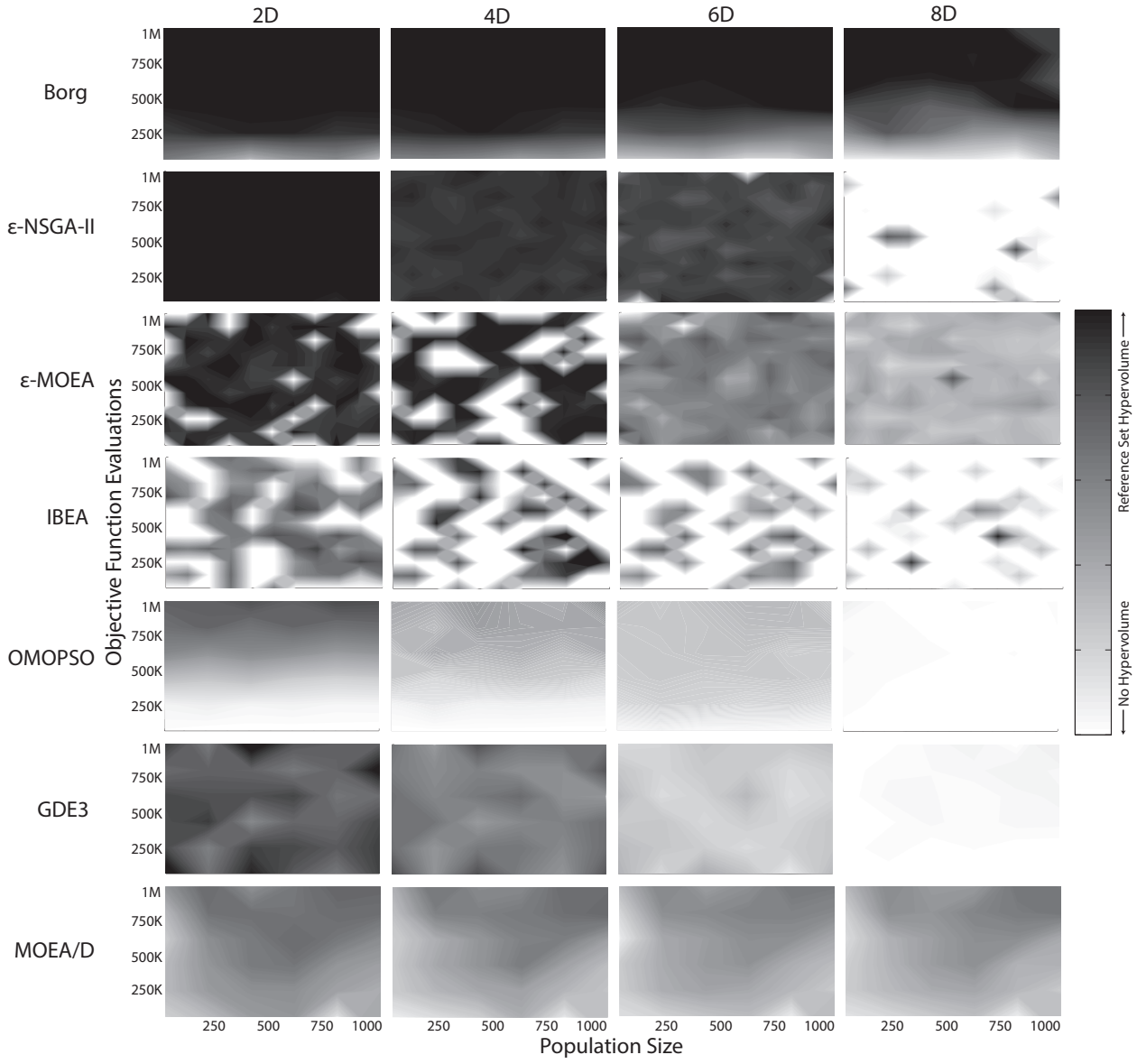
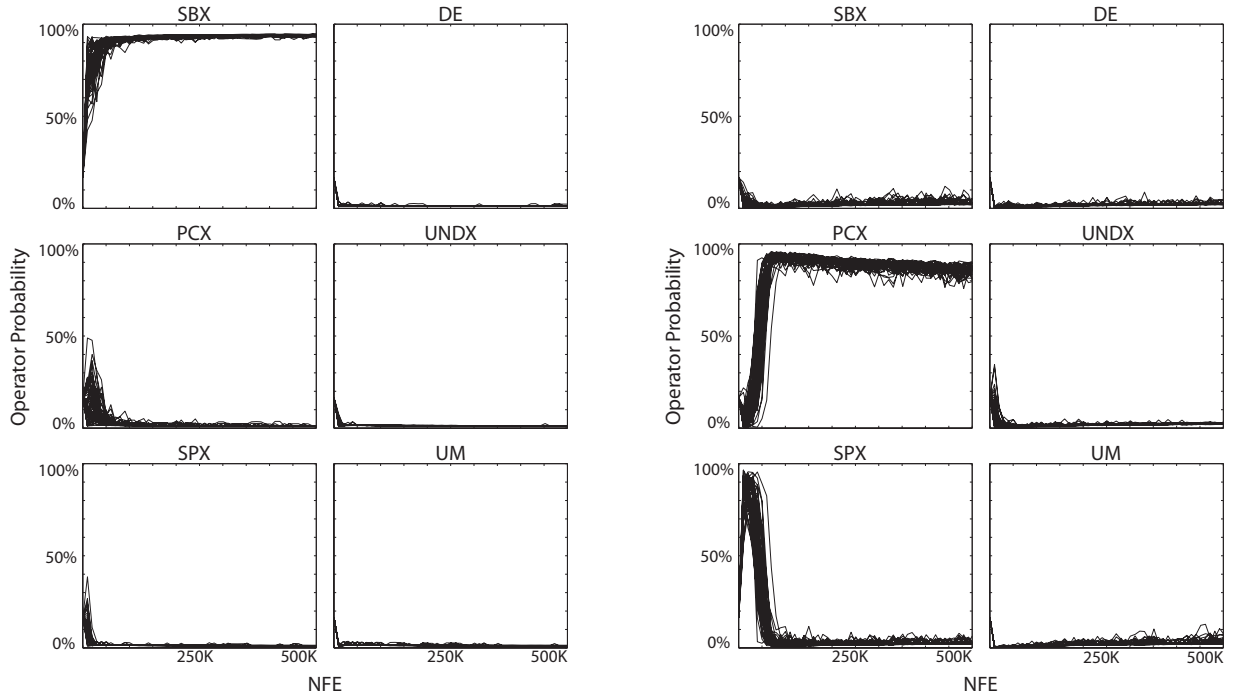


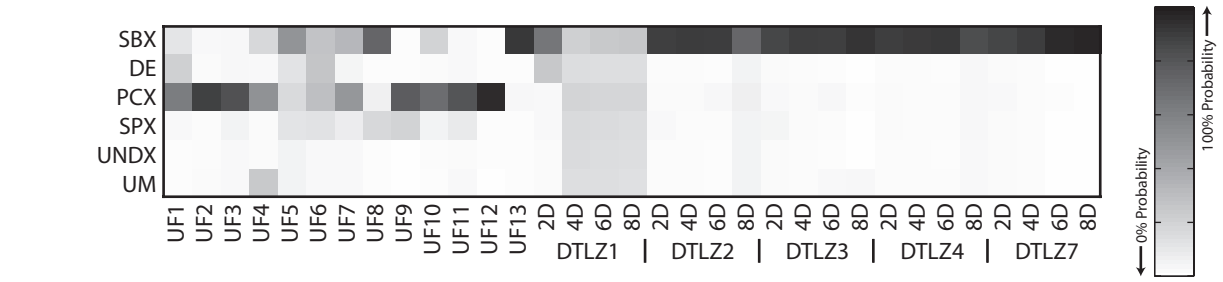
Figure 3.8: Control map showing the relation between population size and number of objective function evaluations on the DTLZ1 problem from 2 to 8 objectives.



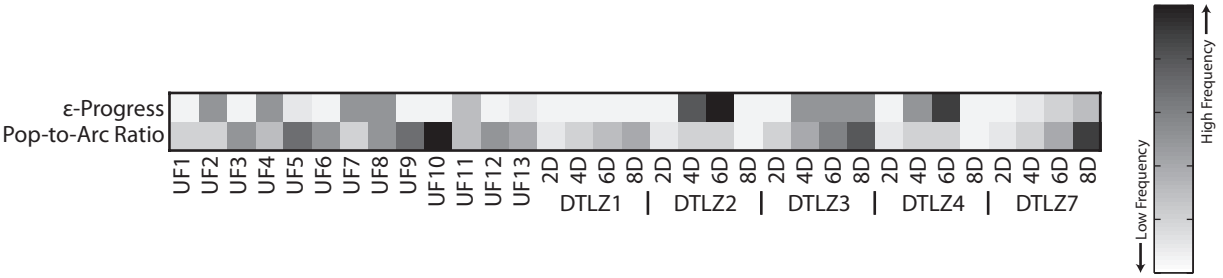
(a) DTLZ2 5D Unrotated

(b) DTLZ2 5D Rotated

Figure 3.9: Depicts the effect of epistasis on success of operators in the Borg MOEA's auto-adaptive multi-operator recombination on an unrotated and rotated instance of the DTLZ2 problem. (a) shows the unrotated version from the DTLZ test suite; (b) shows the rotated version from the CEC 2009 competition.



(a) Operator Probabilities



(b) Restart Frequencies

Figure 3.10: (a) The percentage of operator usage throughout an entire run across all tested problems using a set of fixed parameters. Black cells indicate 100% usage and white cells indicate 0% usage of each operator. SBX and PCX are the two dominant operators on unrotated and rotated problems, respectively, while the other operators show moderate influence on several problems. (b) The restart frequencies due to ϵ-progress and the population-to-archive ratio. ϵ-Progress is scaled so black cells indicate the maximum of 826 restarts observed during any run; the population-to-archive ratio is scaled so black cells indicate the maximum of 14 observed restarts.

3.3.2 Auto-Adaptive Multi-Operator Behavior

Next we demonstrate the ability of the auto-adaptive multi-operator recombination to adapt to a specific problem. Several of the tested problems can be classified into *unrotated* and *rotated* instances. Rotated instances have high degrees of conditional dependence between decision variables. Such conditional dependencies can degrade the performance of recombination operators, but we claim the auto-adaptive multi-operator procedure is capable of identifying and exploiting rotationally-invariant operators on such problems. Figure 3.9 shows the operator probabilities as discussed in Section 3.1.4 throughout the execution of the Borg MOEA on an unrotated and rotated instance of the DTLZ2 problem. The plots show 50 replicates of the Borg MOEA executed with the recommended parameters from Section 3.2.3. As expected, the Borg MOEA correctly selects rotationally-invariant operators to maximize performance on the rotated problems. It is interesting to note in Figure 3.9 that multiple operators work cooperatively during search and that their emphasis in search is highly dynamic (e.g., see SPX and PCX in Figure 3.9b).

A more comprehensive view of operator probabilities is given in Figure 3.10a, which shows the percentage of operator usage throughout an entire run across all tested problem instances. Each cell in the figure is colored according to the percentage of operator use by calculating the "area under the curve" in the plots in Figure 3.9:

$$\text{Percentage}_i = \frac{\sum_j O(i,j)}{\sum_i \sum_j O(i,j)}, \tag{3.4}$$

where $O(i,j)$ is the probability of applying operator i after j objective function evaluations. Using a similar setup as above, the results are averaged over 50 replicates of the Borg MOEA executed with the recommended parameters from Section 3.2.3. Figure 3.10a shows that SBX is dominant on the unrotated DTLZ problems, whereas PCX, SBX and several other operators have significant contributions on the UF problems. This confirms that *a priori* operator selection is non-trivial, especially on real-world problems where the problem characteristics are most likely unknown. Analysis of both Figure 3.6b and Figure 3.10a show that in UF1, UF4, UF7, UF9, UF11 and DTLZ1, the Borg MOEA's high attainment is benefiting from the cooperative utilization of several variational operators. These results are corroborated by the findings of Vrugt and Robinson (2007) and Vrugt et al. (2009), who also show that even operators that provide minor contributions can critically influence the quality of final results.

Figure 3.10b shows the frequency that ϵ-progress and population-to-archive ratio triggered restarts across all tested problem instances. On the DTLZ problem instances, we observe higher frequencies of both ϵ-progress and population-to-archive ratio restarts as the problem dimension is increased. As increasing the problem dimension generally results in proportionally larger non-dominated Pareto sets, the population-to-archive ratio should be triggered more frequently with the increasing archive size. Overall, Figure 3.10 demonstrates that the auto-adaptive multioperator component and the two restart triggers are actively used across a variety of problems. In the following section, we extend this analysis to show that the combination of all three components are necessary for the performance and reliability of the Borg MOEA.

3.3.3 Critical Components of Borg

We conclude this analysis by determining how critical each of individual constituent components of the Borg MOEA are to its overall performance and reliability. The components analyzed are (A) population-to-archive ratio triggered restarts with adaptive population sizing, (B) ϵ-progress triggered restarts, and (C) the auto-adaptive multioperator recombination operator. We repeated the analyses based on 1000 LHS parameterizations, where each parameterization is run for 50 random seed replicates, as done earlier, but with the individual components enabled or disabled to run all 6 potential variants. These variants are compared against the baseline ϵ-MOEA and the full Borg MOEA.

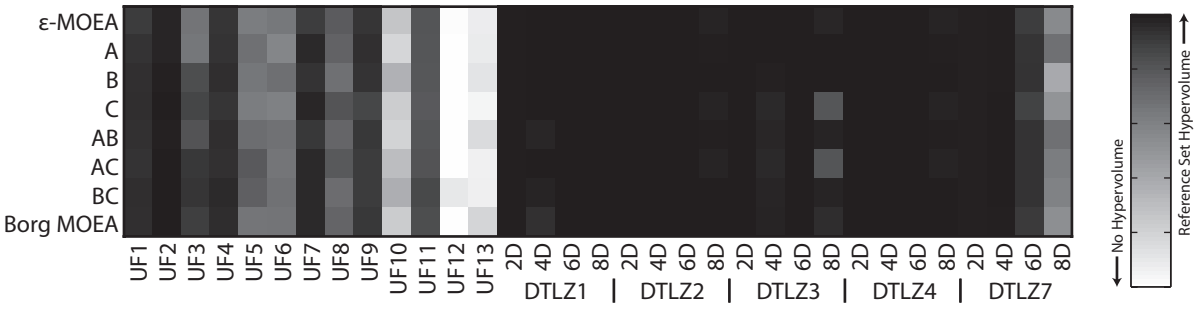
Table 3.2 shows the statistical comparison of the different combinations. On the majority of the tested cases, the full Borg MOEA variant is equivalent or superior. Given that the

Table 3.2: Statistical comparison of the critical components of the Borg MOEA based on the 75% quantile of the hypervolume, generational distance and ϵ_+-indicator metrics. +, =, and − indicate the full Borg MOEA's 75% quantile was superior, statistically indifferent from or inferior to the competing variant, respectively. The enabled components in each variant are identified with letters: (A) population-to-archive ratio triggered restarts with adaptive population sizing; (B) ϵ-progress; and (C) auto-adaptive multioperator recombination.

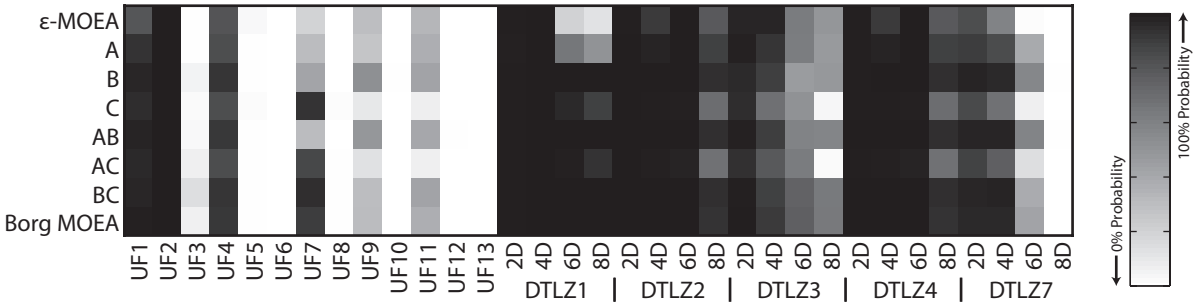
	Hypervolume			Generational Distance			ϵ_+-Indicator		
Variant	+	=	−	+	=	−	+	=	−
A	17	6	10	22	4	7	16	4	13
B	11	10	12	11	4	18	9	8	18
C	21	6	6	24	4	5	15	7	11
AB	11	13	9	11	4	18	9	7	17
AC	20	6	7	23	7	3	14	8	11
BC	5	20	8	2	28	3	4	22	7

test problem suite used in this study is biased towards a few operators (i.e., SBX on DTLZ), it is expected that the variants B and AB are competitive. Since the Borg MOEA must expend objective function evaluations learning the dominant operator(s), the variants using the dominant operator by default have a competitive advantage. The full potential of auto-adaptive multioperator variation is on real-world applications, where the dominant operators are not known *a priori* and are likely to vary. Simply having the ability to discover this operator dependency is a significant contribution and strength of the Borg MOEA.

Figure 3.11 shows the best achieved value and the probability of attaining at least 75% of the reference set hypervolume for the different variants. As expected, ϵ-progress has a very strong impact on proximity, but requires the other operators for diversity, as seen on the darker shadings for variants B, AB and BC in Figure 3.11b. The effects of the auto-adaptive multioperator variation operator can be seen on a number of problems, and is very pronounced on UF7. The variants C, AC, BC and the full Borg MOEA show significant improvements on UF7. From Figure 3.10a, we see this was achieved by identifying SBX and PCX as dominant operators. Figure 3.11 does verify that the variants without the multioperator learning curve do have an advantage on the DTLZ test problems in which SBX is a dominant operator. The population-to-archive ratio triggered restarts with adaptive population sizing appear to have a more pronounced effect on the higher dimensional DTLZ instances. This distinction is clearly visible when comparing ϵ-MOEA with variant A. The earlier results seen on ϵ-NSGA-II also support the positive impacts of adaptive population sizing as captured in Table 3.2, which shows ϵ-NSGA-II as one of the top performers overall. It can therefore be concluded that all three constituent components of the Borg MOEA contribute to its observed success and its intended use in many-objective real-world applications.



(a) Best Achieved



(b) Attainment

Figure 3.11: Best achieved and 75% attainment results from the analysis of the critical components of the Borg MOEA. (a) shows the best value achieved by the configuration across all seeds, where black indicates values near the reference set hypervolume. (b) shows the probability of attaining at least 75% of the reference set hypervolume for each problem. Black indicates 100% probability; white indicates 0% probability. The enabled components in each variant are identified with letters: (A) population-to-archive ratio triggered restarts with adaptive population sizing; (B) ϵ-progress; and (C) auto-adaptive multioperator recombination.

3.4 Conclusion

The Borg MOEA provides robust optimization by assimilating design components from other MOEAs and introduces several novel features. ϵ-Dominance archives and ϵ-progress maintain a well-spread Pareto front and monitor convergence speed. If the convergence speed declines and search stagnates, randomized restarts are triggered which revive search by resizing and diversifying the population while carefully maintaining selection pressure. Lastly, the Borg MOEA provides a facility to incorporate multiple recombination operators and automatically adapts its use of these operators based on their relative performance.

Our comparative study demonstrates the efficacy of the Borg MOEA on 33 test problem instances ranging from 2 to 8 objectives. Using a large-scale Latin hypercube sampling of each algorithm's parameters, we observed that the Borg MOEA outperformed the competing algorithms on the majority of the test problems. The Borg MOEA reliably and consistently produced Pareto sets matching or exceeding the best-known algorithms in terms of hypervolume, generational distance and ϵ_+-indicator.

In particular, the Borg MOEA showed significant advantages over competing algorithms on many-objective, multi-modal problems. On such problems, the Borg MOEA produced results with significantly better hypervolume and achieved such results with higher probability. However, all of the tested algorithms showed serious reliability issues on several UF problems, an issue which should elicit further investigations.

While the Borg MOEA's use of multiple recombination operators requires users to set more parameters, our control map results highlight that the Borg MOEA's auto-adaptive search features strongly reduce parameterization challenges and provide large "sweet spots," even on problems with 8 objectives. Nonetheless, operator selection and parameterization are important considerations when maximizing the performance of any MOEA. In Chapter 4, we detail the dependencies between search operators, their parameters and many-objective performance for a broad range of MOEAs, including the Borg framework introduced in this study.

Chapter 4

Framework for Diagnosing Search Controls and Failure Modes

This chapter is drawn from the paper: "Hadka, D. and Reed, P. (2012). *Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization.* Evolutionary Computation, 20(3):423-452."

To date, the complex dynamics of MOEAs when solving many-objective optimization problems has limited the analytical assessment of their strengths and weaknesses. Alternatively, with the advent of the DTLZ (Deb et al., 2001), WFG (Huband et al., 2006) and CEC 2009 (Zhang et al., 2009b) test problem suites, the systematic study of objective scaling through numerical experimentation has provided important insights into MOEA scalability for increasing objective dimensions. Khare et al. (2003) published the first study examining the effects of objective scaling on proximity and diversity using four DTLZ problems. Several additional experimental studies have been published using fixed or tuned parameters, as shown in Table 4.1. Purshouse and Fleming (2003, 2007) published the first study constructing control maps across a range of problem dimensions for the recombination and mutation operators for the Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2000) by sampling points on a grid from parameter space. They demonstrated that the parameterization "sweet spot" migrates as the number of objectives increases. This result suggests that default parameter settings commonly used in the literature are not applicable to problems of varying dimensions.

More generally, Goh and Tan (2009) discuss the challenges in designing frameworks for the empirical analysis and performance assessment of MOEAs. They assert three important design requirements for any diagnostic framework: (1) multiple performance metrics covering the functional objectives of multiobjective optimization, (2) an adequate sample of problems, and (3) the ability to uncover pertinent parameter controls and dynamic search behavior within the algorithm. This study introduces a systematic framework for diagnosing the search capabilities of MOEAs while providing guidance on how the key multivariate interactions between an algorithm's parameters and its operators change as the number of objectives increases. This study represents one of the largest and most comprehensive computational experiments ever performed on MOEAs. Millions of algorithm runs using trillions of fitness

Table 4.1: List of prior comparison studies analyzing objective scaling for MOEAs. † marks algorithms modified specifically for handling many-objective optimization.

Algorithms	Problems	Objectives	Parameters	Reference
NSGA-II, SPEA2, PESA	DTLZ 1-3, 6	2-8	Tuned	Khare et al. (2003)
NSGA-II, MSOPS, RSO	Custom	2, 4, 6	Fixed	Hughes (2005)
NSGA-II, POGA†	DTLZ 1-4, 6-8	4-8	Tuned	di Pierro (2006)
NSGA-II, SPEA2, IBEA	DTLZ 1-7, WFG	2-4	Fixed	Wagner et al. (2007)
NSGA-II	DTLZ 1-3, 6	4-8	Tuned	Praditwong and Yao (2007)
NSGA-II, SPEA2, ϵ-MOEA	DTLZ 1-2	3-6	Fixed	Wagner et al. (2007)
NSGA-II, POGA	DTLZ 1-7	4-8	Tuned	di Pierro et al. (2007)
NSGA-II	DTLZ 2	3, 6, 12	Grid	Purshouse and Fleming (2003, 2007)
PESA-II	NK Landscapes	2, 5, 10	Fixed	Knowles and Corne (2007)
NSGA-II†	Knapsack	2, 4, 6, 8	Fixed	Ishibuchi et al. (2008a)
NSGA-II†	DTLZ2	6, 8, 12	Fixed	Adra and Fleming (2009)

function evaluations were executed to explore the design space of state-of-the-art MOEAs. Such extensive experimentation supports the comparison of each algorithm's best achieved metric values, their probabilities of attaining high-quality approximation sets, efficiency, and controllability without biasing results to "tuned" rules for parameterization. Failures in this study for the first time imply failures in the MOEA's design — selection, variation operators, ranking, diversity maintenance, archiving, etc. and their interactions — rather than the synoptic analysis of poor parameterization effects, which has been the dominant focus of prior literature.

The diagnostic framework and our proposed measure of controllability are described in detail in Section 4.1. The results of a comparative analysis using this diagnostic framework is presented in Section 4.2 along with an analysis of search controls and failure modes. This paper concludes in Section 4.3 with a discussion of the impact of this work.

4.1 Diagnostic Framework

The primary contribution of this chapter is a diagnostic framework for robustly comparing how MOEA operators, their parameterization, and the interactions between these factors influence their successes and failures in many-objective optimization. Section 4.1.1 defines the best attained approximation set, probability of attainment, efficiency, and controllability metrics used by this diagnostic framework. Section 4.1.2 introduces variance decomposition of controls for analyzing the multivariate interactions between parameters. Section 4.1.3 outlines the computational experiment performed in this study. Table 4.2 identifies common notations used throughout this section.

4.1.1 Search Control Metrics

Whereas the performance metrics discussed in Section 2.8 compare the quality of approximation sets from single runs, they are only applicable to fixed parameterizations. In this study we propose instead the following metrics for statistically sampled ensembles of approxima-

Table 4.2: Notation used in study.

Symbol	Description
M	Number of objectives
$s \in S$	A seed from the set of random number generator seeds
$p \in \mathcal{P}$	A parameter set in the parameter block
A	One of the studied algorithms
A_p^s	A single run of the algorithm A using parameter set p and seed s; returns an approximation set
A_p	Shorter version of A_p^s implying a single seed s is used
$\mathcal{M}(A_p^s)$	Performance metric applied to the approximation set from a single run
$\mathcal{M}^*$	Target metric value (i.e., best achievable metric value given a reference set)

tion sets and their corresponding performance metrics to provide guidance on an MOEA's utility. Our diagnostic framework classifies an MOEA's utility using four measures: best achieved value, probability of attainment, efficiency, and controllability.

Best Achieved Value The majority of studies report the best achieved end-of-run performance metric value. However, unlike the majority of studies where results are based on fixed or tuned parameters, our best attained result is drawn from a large statistical sampling of the full feasible parameterization ranges for all of the major operators in each algorithm in order to provide a rigorous measure of an MOEA's best performance.

$$\text{Best Achieved Value} = \max_{p \in \mathcal{P}} \mathcal{M}(A_p) \tag{4.1}$$

Probability of Attainment While the best achieved value is an absolute measure of an MOEA's search quality, the reliability of an algorithm is a stronger indicator of an MOEA's utility. This is particularly important on rotated, multi-modal, many-objective problems where an MOEA may be capable of producing quality end-of-run approximation sets, but the probability of doing so is low. We propose measuring an MOEA's reliability with the probability that the end-of-run approximation set surpasses an attainment threshold, α. From the set of parameters $\mathcal{P}$, the set of parameters surpassing this attainment threshold is

$$\mathcal{P}^\alpha = \{p \in \mathcal{P} : \mathcal{M}(A_p) \geq \alpha\}. \tag{4.2}$$

From this, the probability of attainment is defined by

$$\text{Probability of Attainment} = \frac{|\mathcal{P}^\alpha|}{|\mathcal{P}|}. \tag{4.3}$$

Efficiency MOEAs that achieve high attainment probabilities with fewer objective function evaluations are preferred over those that require more time to search. Efficiency measures the minimum number of objective function evaluations (NFE) required to achieve a

high probability of attainment. Given a range R of NFE values, we define a band of statistically sampled parameterizations within that range as

$$\mathcal{B}_R = \{p \in \mathcal{P} : \text{NFE}(p) \in R\}, \tag{4.4}$$

and the subset of parameterizations in that band surpassing the attainment threshold as

$$\mathcal{B}_R^\alpha = \{p \in \mathcal{B}_R : \mathcal{M}(A_p) \geq \alpha\}. \tag{4.5}$$

Efficiency is defined as the minimum NFE band R such that 90% of the parameters in the band surpass the attainment threshold:

$$\text{Efficiency} = \min \left\{ R : \frac{|\mathcal{B}_R^\alpha|}{|\mathcal{B}_R|} \geq 0.9 \right\}, \tag{4.6}$$

where $R = [\Delta i, \Delta(i+1)]$ for $i = \{0, \ldots, 99\}$ and $\Delta = 10000$. Note the similarities between these equations and those for the probability of attainment. The choice of 90% is based on our efforts to maintain consistency and rigor across our performance measures. In the context of this specific study, there were no significant differences in efficiency if 50% and 75% thresholds were stipulated.

Controllability Lastly, we are interested in the distribution of the parameters in $\mathcal{P}^\alpha$. Controllable algorithms are those which exhibit "sweet spots", or regions in parameter space with high attainment probabilities. The *correlation dimension* (Grassberger and Procaccia, 1983) of $\mathcal{P}^\alpha$ is our measure of controllability. Hence, controllability is computed by

$$\text{Controllability} = \lim_{r \to 0} \frac{\ln(C(r))}{\ln(r)}, \tag{4.7}$$

where

$$C(r) = \frac{1}{N(N-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{N} H(r - |p_i - p_j|) \tag{4.8}$$

with $p_i, p_j \in \mathcal{P}^\alpha$, $N = |\mathcal{P}^\alpha|$ and H is the Heaviside function defined by

$$H(u) = \begin{cases} 0 & \text{if } u < 0 \\ 1 & \text{if } u \geq 0. \end{cases} \tag{4.9}$$

Conceptually, $C(r)$ is the average fraction of parameter sets within a radius r of each other. The growth of $C(r)$ with respect to r reflects dimensionality since higher dimensional spaces permit more opportunities for points to be close (Baker and Gollub, 1990). As shown in Figure 4.1, rather than computing (4.7) directly, it is recommended to instead compute the slope where the correlation dimension estimate $\ln(C(r))/\ln(r)$ is relatively constant (this region is called the *plateau region* in the literature) (Nayfeh and Balachandran, 1995).

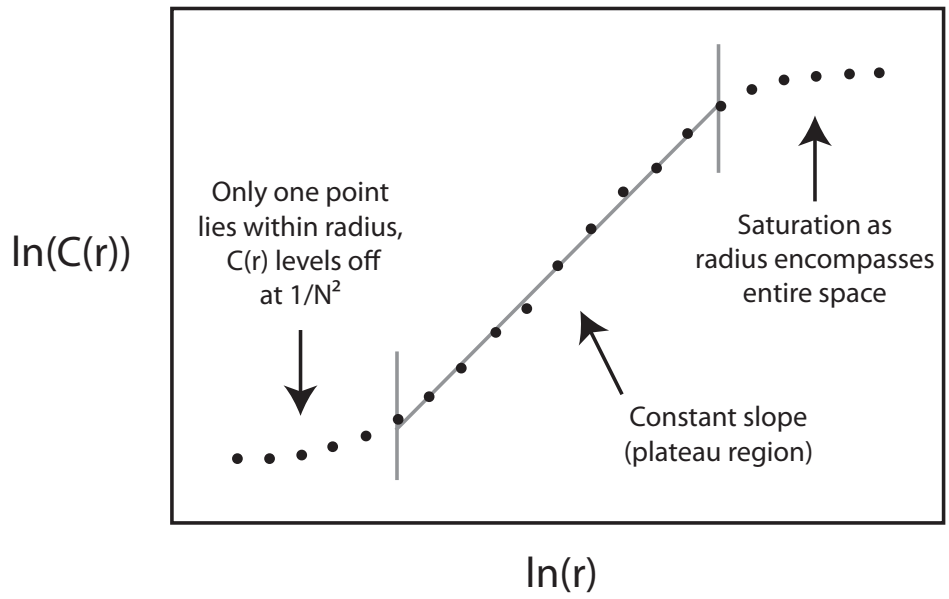


Figure 4.1: The correlation dimension is the slope where the correlation dimension estimate $\ln(C(r))/\ln(r)$ is relatively constant (this region is called the *plateau region* in the literature). As indicated, small and large radii do not reflect dimensionality.

To compute (4.7), the effective parameters $\mathcal{P}^\alpha$ are first normalized to reside within the unit hypercube. The $N(N+1)/2$ pairwise distances between effective parameters are computed and stored in an array. $C(r)$ from (4.8) is computed for various $r \in [0,1]$ by referencing distances in this stored array. Next, the plateau region is identified, as shown in Figure 4.1. Let $\mathcal{R} = \{r : r_{\min} \leq r \leq r_{\max}\}$ be the sampled values of r within some bounds. The linearity of $\ln(C(r))$ versus $\ln(r)$ is determined by computing the correlation coefficient (Edwards, 1993)

$$\rho = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{\sqrt{n\left(\sum x^2\right) - \left(\sum x\right)^2}\sqrt{n\left(\sum y^2\right) - \left(\sum y\right)^2}}, \tag{4.10}$$

where the summations are over the values $r \in \mathcal{R}$, $n = |\mathcal{R}|$, $x = \ln(r)$ and $y = \ln(C(r))$. Searching for the largest bounds, $r_{\max} - r_{\min}$, with $|\rho| \geq 1 - \xi$ identifies the plateau region. This study used $\xi = 0.001$ to ensure a high degree of linearity. Finally, the slope of the identified plateau region and the estimation of (4.7) is calculated using linear least squares regression (Edwards, 1993)

$$m = \frac{n\left(\sum xy\right) - \left(\sum x\right)\left(\sum y\right)}{n\left(\sum x^2\right) - \left(\sum x\right)^2}, \tag{4.11}$$

with the same variables as (4.10).

In summary, controllability measures the correlation between effective parameters. Thus, larger controllability values indicate increasingly larger perturbations to an effective parameter set will still result in good performance, which indicates the existence of "sweet spots".

The existence of "sweet spots" is necessary for the effective control of search via parameterization. Without "sweet spots", adapting parameters becomes hard since effective parameters are like needles in a haystack — small perturbations to effective parameters will likely result in poor performance.

4.1.2 Variance Decomposition of Controls

The highly non-linear nature of MOEAs emerges from complex interactions between their operators and their parameterization, which has limited the analysis of generalized MOEA behavior. Most studies to date only examine one or two parameters in isolation (Harik and Lobo, 1999). However, recent advances in sensitivity analysis have introduced techniques for computing all parameter effects and their multivariate interactions more reliably and with fewer parametric assumptions relative to traditional methods like analysis of variance (ANOVA).

Variance decomposition attributes to each parameter the percentage it contributes to an output ensemble's variance. *First-order* effects represent variation caused solely by a single parameter. *Second-order* and *higher-order* interaction effects represent variation caused by two or more parameters in conjunction. *Total-order* effects represent for each parameter the summation of its first-order and all higher-order effects.

While ANOVA has been traditionally used to capture first- and second-order effects, the variance decomposition method developed by I.M. Sobol' with modifications by Saltelli et al. (2008) provides many advantages. First, using the implementation in Saltelli et al. (2008), the total-order effects can be computed with little additional cost over Sobol's original implementation. Second, whereas uniform random sampling of parameters yields a sampling error growth rate of $1/\sqrt{N}$, sampling parameters with Sobol's quasi-random sequence generator yields an error growth rate of $1/N$, a significant improvement in convergence (Tang et al., 2007). In this study, $N = |\mathcal{P}|$. Third, the rank-ordering of parameters by Sobol's method has been observed in practice to be more reliable and stable than ANOVA (Tang et al., 2007). Finally, Sobol's method is model independent and only assumes parameter independence. ANOVA, on the other hand, assumes normally-distributed model responses, homoscedasticity, and independence of cases (Tang et al., 2007).

For these reasons, Sobol's variance decomposition is used in this study to identify an MOEA's key parameters and investigate the multivariate interactions between its control parameters. Error estimates are determined using bootstrapping. A more detailed discussion of Sobol's variance decomposition and bootstrapping is provided in Appendix B.

4.1.3 Computational Experiment

This study applies the Borg MOEA and the eight MOEAs listed in Section 2.6 to the 33 test problem instances listed in Table 2.1. Figure 4.2 depicts the overall outline of this computational experiment, which is described in detail below. To permit Sobol's variance decomposition for each algorithm, a *parameter block* consisting of $1000(2P + 2)$ parameter sets is generated using a Sobol' sequence-based statistical sampling method, where P is the

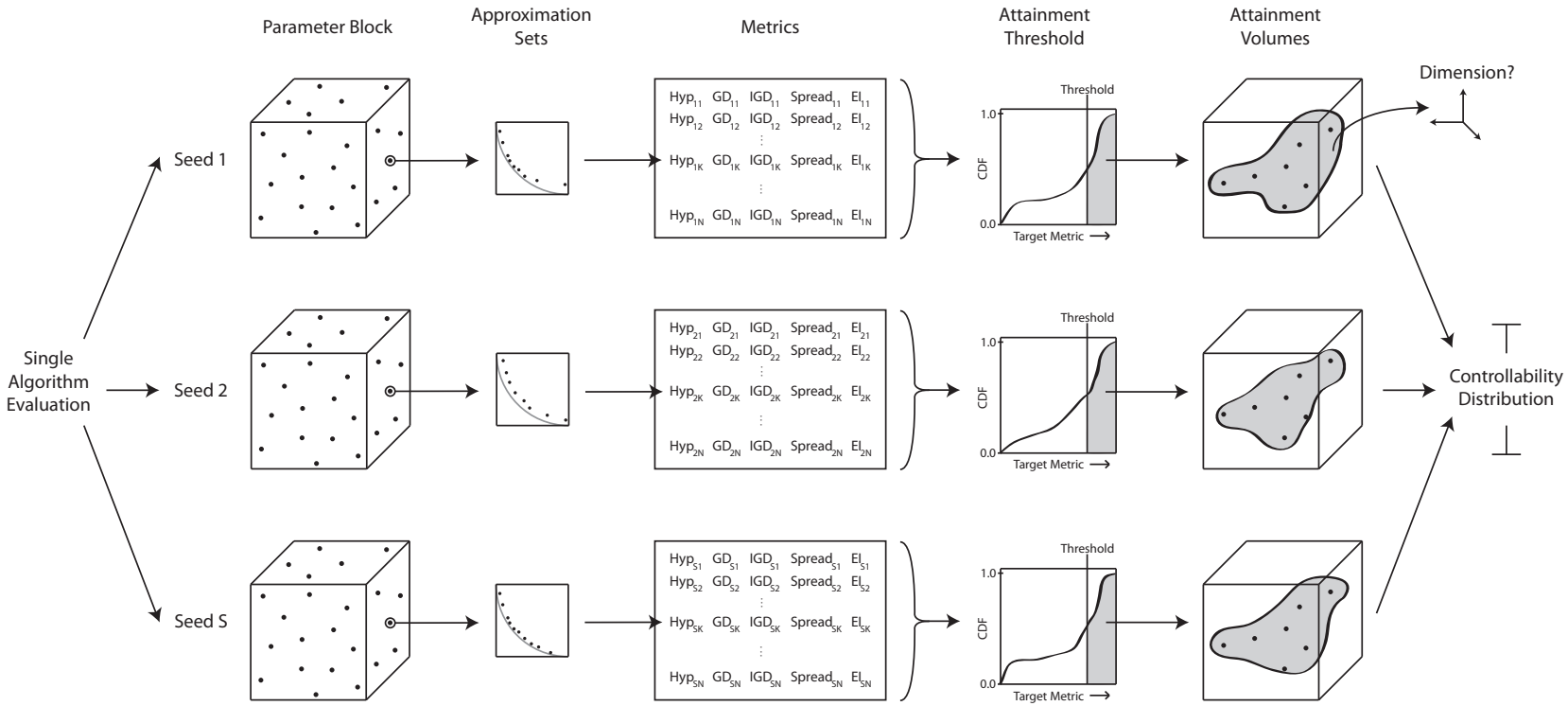


Figure 4.2: For each algorithm, a Sobol' sequence-based statistical sampling of its parameters is generated (i.e., the parameter block). Each parameter set in the parameter block is evaluated using multiple random number seed trials ($S = 50$) to improve the statistical quality of our results. From the resulting non-dominated approximation sets, the corresponding performance metrics are computed. An attainment threshold retains all parameter settings surpassing the threshold value, which are then used to compute the probability of attainment, efficiency, and controllability measures.

number of parameters controlling the algorithm. For each parameter set in the parameter block, the algorithm is run 50 times using different initial pseudo-random number generator seeds for each problem instance. The same parameter block is used across all seeds and problem instances for each algorithm. The result of each run is a Pareto approximation set which is evaluated using the performance metrics discussed in Section 2.8. The multiple random number seed trials render the results independent of the initial population and improve the statistical quality of our results.

After all the data is collected, the search control metrics and variance decomposition of controls are computed. Each parameter block is analyzed to identify only those runs surpassing a 75%-attainment threshold relative to the known reference sets. The resulting *attainment volume* is used to compute the probability of attainment, efficiency, and controllability search control metrics. Along with the best achieved value, these measures of algorithmic utility can be used to make observations of the current state-of-the-field for solving many-objective problems. Additionally, our framework utilizes Sobol's variance decomposition to rigorously assess each algorithm's search controls while simultaneously providing insights into the multivariate interactions between parameters and operators. Our proposed use of variance decomposition thus characterizes the effect of objective scaling on MOEA search.

The range of sampled parameter values is taken from Hadka and Reed (2012a). The number of fitness evaluations was sampled between $[10000, 1000000]$ in order to permit tractable execution times while providing meaningful results. The population size, offspring size, and archive sizes are all sampled between $[10, 1000]$. This range was chosen to encompass the commonly employed "rule-of-thumb" population sizes in MOEA parameterization recommendations. Mutation rate, crossover rate, and step size encompass their entire feasible ranges of $[0, 1]$. Distribution indices for SBX and PM range between $[0, 500]$, which is based on the "sweet spot" identified by Purshouse and Fleming (2007).

The experiments were executed on the CyberSTAR computing cluster at the Pennsylvania State University, which consists of 512 2.7 GHz processors and 1536 2.66 GHz processors. In total, 280 million algorithm runs were executed requiring approximately 225 years of computational effort. To the best of our knowledge, this is the most extensive and comprehensive comparison study of MOEAs to date. Consequently, our results do not rely on fixed or tuned parameters and provides a state-of-the-field baseline for many-objective evolutionary optimization. While the computational expenditure for this study is high, it has freed our analysis and results from restrictive assumptions, and is the first robust analysis that statistically samples the design space of MOEAs.

4.2 Results and Discussion

Figures 4.3, 4.4, 4.5, and 4.6 show the best achieved value, probability of attainment, efficiency and controllability measures, respectively, for the 33 test problem instances. Each plot contains three horizontal subplots showing the generational distance (GD), hypervolume, and ϵ_+-indicator performance metrics. Each subplot is composed of shaded squares corresponding to the problem (x-axis) and the algorithm (y-axis). The interpretation of the

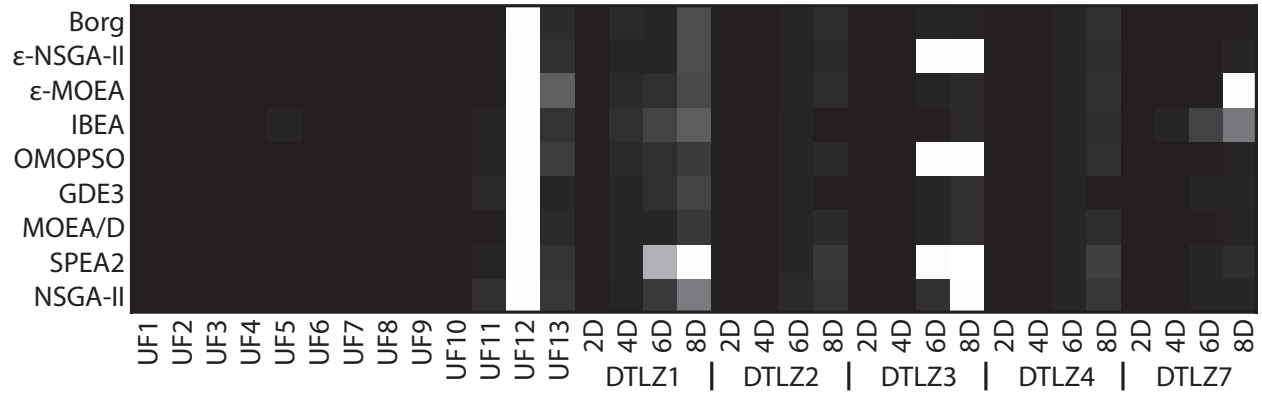
shading depends on the individual plot, but in all cases black represents the ideal result and white the worst result. All shadings are scaled linearly as indicated in the legends.

Figure 4.3 shows for each MOEA its overall best achieved metric value for the three performance metrics. Dark regions indicate at least one of the sampled parameter sets attained performance metric values very near to the target metric value. Starting with GD, which measures the average distance from objective vectors in the approximation set to the nearest neighbor in the reference set, we observe that at least one parameter set was able to attain near-optimal convergence to the reference set for most problem instances. We observe that all of the algorithms had difficulty on the UF12 problem from the CEC 2009 test suite, and ϵ-NSGA-II, OMOPSO and SPEA2 had difficulty on the 6 and 8 dimension cases of the DTLZ3 problem. In addition, NSGA-II struggled on the 8D DTLZ3 instance and SPEA2 struggled on the 8D DTLZ1 instance. This indicates that apart from these few exceptions, the majority of the tested algorithms are capable of producing at least one approximation set in close proximity to the reference set. While GD measures proximity to the reference set, a non-diverse population covering only a small fraction of the reference set can receive near-optimal GD values. In other words, GD provides no information about diversity.

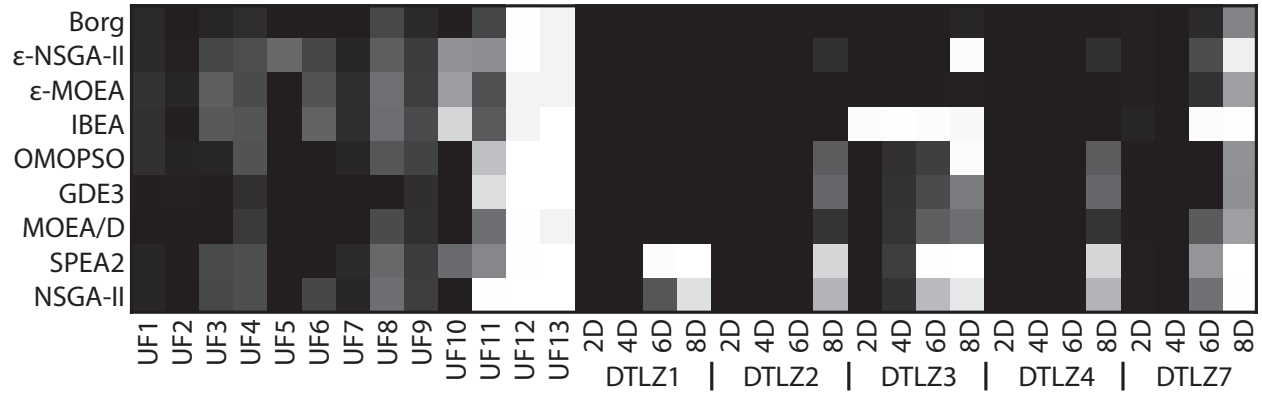
The hypervolume performance metric, which measures the volume of space dominated by the approximation set, combines proximity and diversity into a single evaluation metric. Again, the majority of the tested algorithms are able to generate at least one approximation set with a hypervolume near the reference set. First, we observe low hypervolume values on UF11, UF12 and UF13. Given the near-optimal GD values on UF11 and UF13, this indicates the MOEAs struggle to maintain a diverse set of solutions on these problem instances. This loss in diversity is also apparent for IBEA on DTLZ3 and DTLZ7. On DTLZ3, IBEA struggles to maintain a diverse approximation set regardless of problem dimension. This indicates a significant search failure for IBEA, particularly given the fact that IBEA is based on the hypervolume indicator. The Borg MOEA is able to achieve near-optimal hypervolume values for the majority of the tested problem instances, only struggling on UF12, UF13 and 8D DTLZ7. The ability of the Borg MOEA to maintain a diverse approximation set is aided by its use of an ϵ-dominance archive.

The last metric shown in Figure 4.3 is ϵ_+-indicator. The ϵ_+-indicator highlights the existence of gaps in the Pareto fronts (i.e., consistency as illustrated in Figure 2.12). The ϵ_+-indicator highlights the difficulty of UF12 and UF13 as detected by GD and hypervolume. A clear pattern emerges on the DTLZ problems showing a degradation in performance of the algorithms at higher problem dimensions. The Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and MOEA/D show a slight advantage, particularly on higher-dimensional DTLZ problem instances.

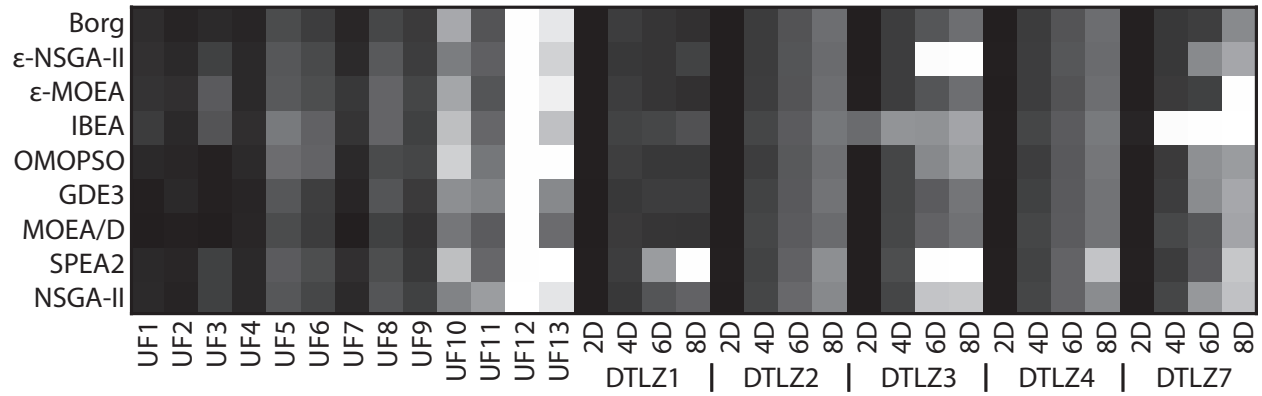
Combining these three performance metrics provides a clear indication as to the quality of an approximation set. A favorable GD value implies good proximity, a favorable hypervolume implies good diversity with proximity, and a favorable ϵ_+-indicator value implies good consistency (i.e., the absence of poorly approximated regions in the approximation set). As an example, an MOEA exhibiting good GD but poor ϵ_+-indicator values implies some regions of the reference set are approximated poorly. Tradeoffs between the various algorithms with



(a) Generational Distance (GD)



(b) Hypervolume



(c) ϵ_+-Indicator

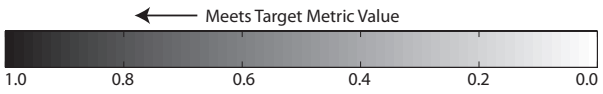


Figure 4.3: The overall best performance for each algorithm on each problem instance is illustrated as the percentage of target metric value achieved. The targets for each problem are based on their true reference sets. Black regions indicate there exists at least one parameter set that yielded near-optimal metric values. White regions indicate no such parameter set exists.

respect to the functional objectives of MOEAs is evident; however, the Borg MOEA shows the most successful results across all functional objectives. Alternatively, IBEA, SPEA2 and NSGA-II struggled to produce diverse approximation sets on many-objective problems.

Readers should note that in addition to the tested algorithms, random search was used to establish a baseline comparison. The random search baseline was established by randomly generating the same number of solutions as were evaluated by the MOEAs and adding them to an ϵ-dominance archive using the same ϵ values as the Borg MOEA and OMOPSO. The performance metrics were computed for the approximation sets generated by random search. In all cases excluding UF12, where all algorithms failed, the MOEAs outperformed random search. This fact is important as it implies the MOEAs are performing non-trivial search.

It is interesting to note the difficulty observed on UF12. UF12 is the rotated version of the 5D DTLZ3 problem originally used in the CEC 2009 competition (Zhang et al., 2009b). This suggests that state-of-the-art MOEAs still show significant search failures on rotated, multi-modal, many-objective problems. This highlights the need for further advancements in this area.

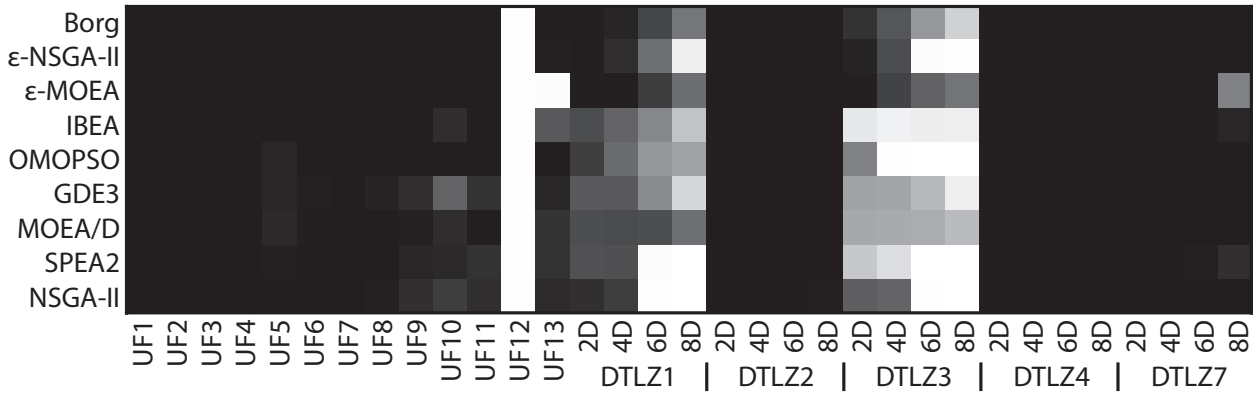
Many studies feature the best observed metric, but such cherry picking of parameters poorly reflects a user's ability to utilize an MOEA in real-world applications where search failures can have actual economic costs. Recall that this study uses an 75%-attainment threshold when calculating the probability of attainment. The probability of attainment, which is the percentage of sampled parameter sets that are able to achieve 75% of each problem instance's reference set, is shown in Figure 4.4. Black identifies cases where the majority of the parameter sets sampled are successful in attaining high quality approximation sets.

Starting with GD in Figure 4.4, we observe that all algorithms exhibit high attainment probabilities on most UF problems and all tested dimensions of DTLZ2, DTLZ4 and DTLZ7. For these cases, the majority of the parameters sampled produce results with a high level of proximity. However, this does not hold for DTLZ1 and DTLZ3. The majority of the tested MOEAs show low attainment probabilities, even on 2D and 4D DTLZ3. The Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and NSGA-II were the only MOEAs that retained high attainment probabilities on 2D and 4D DTLZ3.

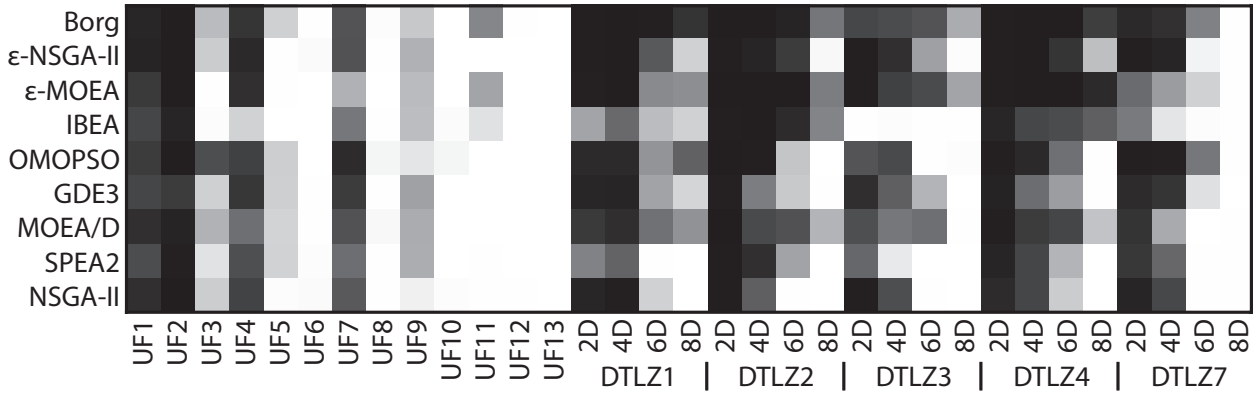
In addition, the hypervolume and ϵ_+-indicator values show diversity and consistency are issues. With the exceptions of UF1, UF2, UF4, UF7 and lower-dimensional DTLZ problem instances, the tested algorithms were not reliably capable of producing well-spread and consistent approximation sets. The Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and MOEA/D provide better diversity and consistency than the other MOEAs, but even these struggle on higher-dimensional instances.

The general trend across all of the algorithms' low attainment probabilities on DTLZ1 and DTLZ3 suggests multi-modal problems can cause significant search failure. In combination, Figure 4.3 and Figure 4.4 show that these algorithms can attain high quality solutions, but the probability of it occurring using commonly selected parameters decreases significantly as the objective space dimension increases.

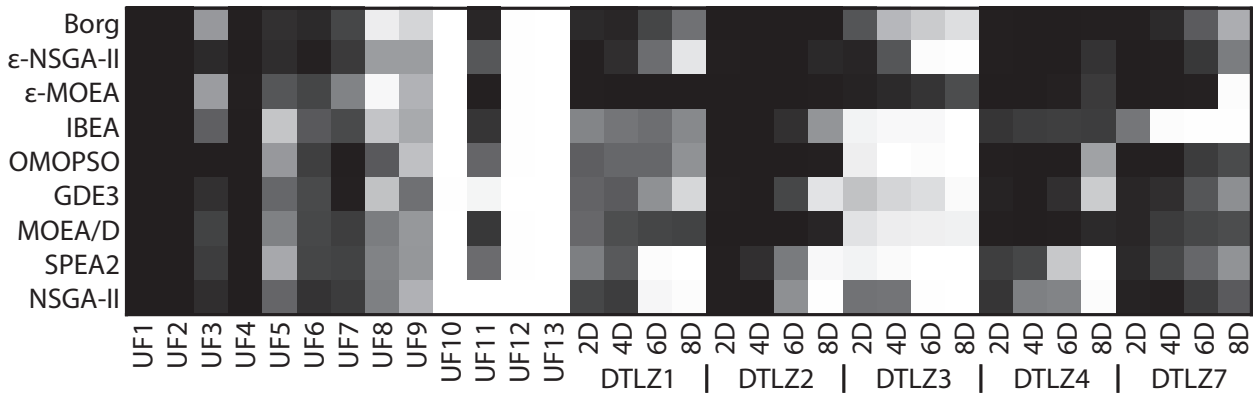
Efficiency reflects the amount of effort expended by the MOEA, in terms of the number



(a) Generational Distance (GD)



(b) Hypervolume



(c) ϵ_+-Indicator

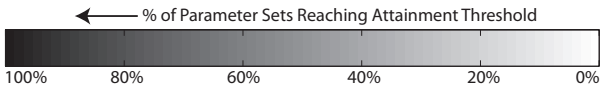
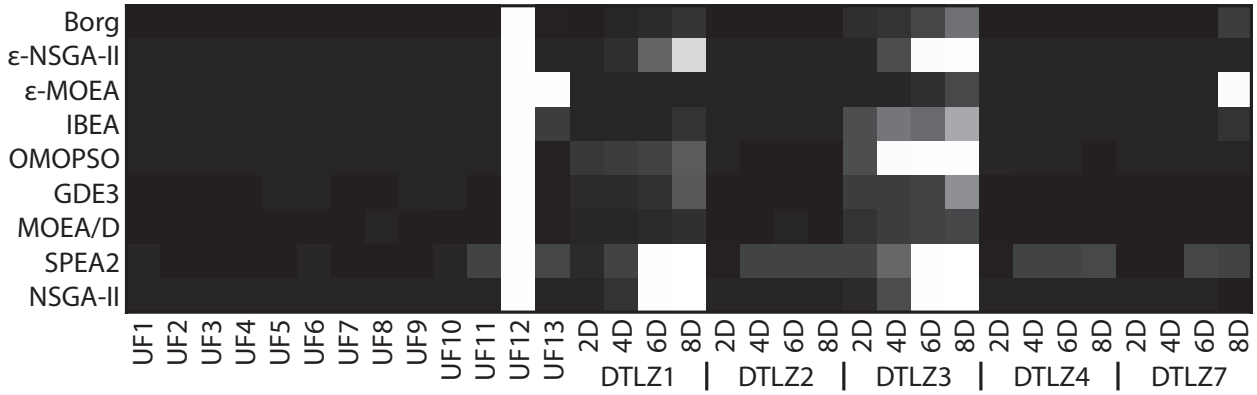
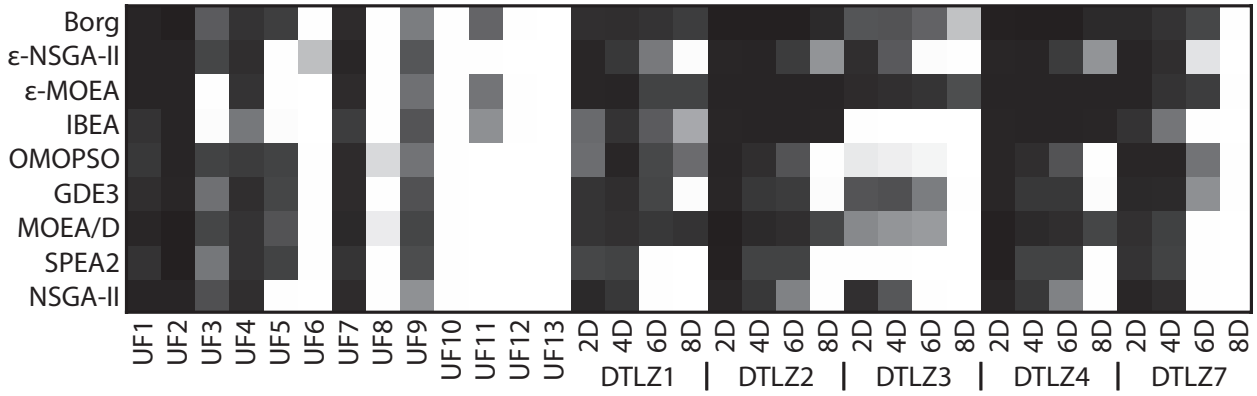


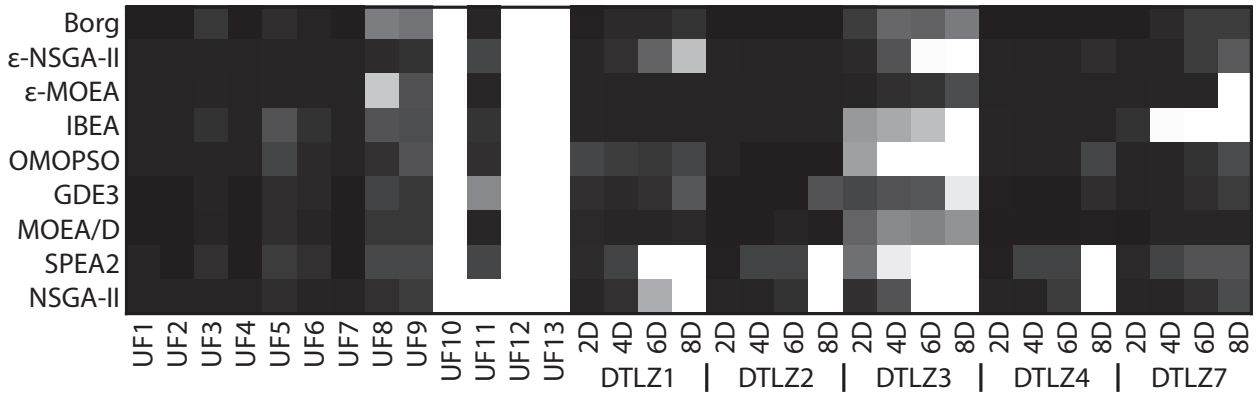
Figure 4.4: The probability of attainment results illustrate the percent of parameter sets for each algorithm that yielded end-of-run metric values surpassing a 75%-attainment threshold. Black regions indicate large success rates while white regions indicate low success rates.



(a) Generational Distance (GD)



(b) Hypervolume



(c) ϵ_+-Indicator

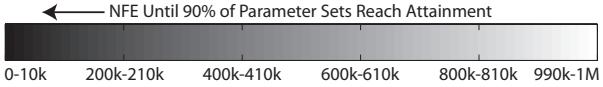


Figure 4.5: The efficiency of each MOEA shows the minimum number of NFE required for the algorithm to reliably (with 90% probability) produce approximation sets surpassing the 75% attainment threshold. Black regions indicate efficient algorithms requiring fewer objective function evaluations. White regions indicate cases where the algorithm failed to surpass the attainment threshold given a maximum of 1000000 evaluations.

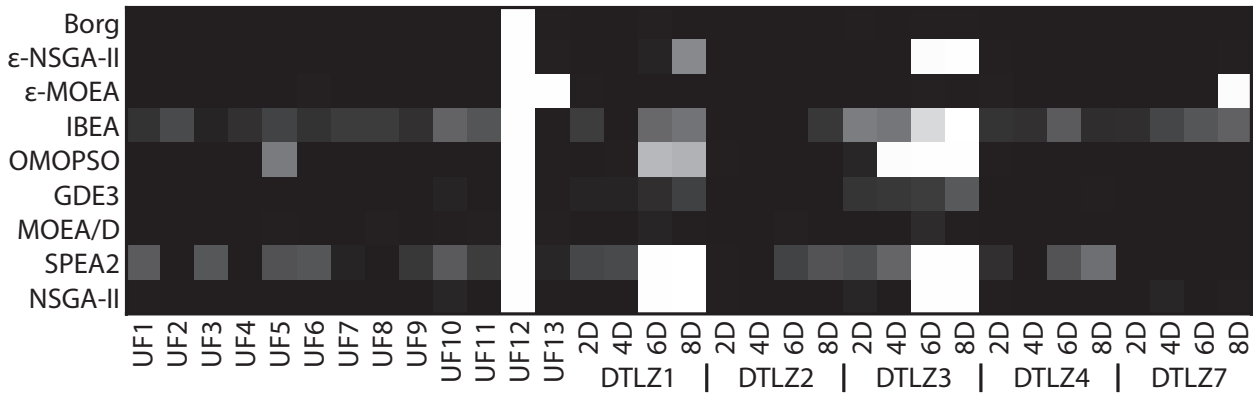
of objective function evaluations (NFE), to produce approximation sets surpassing the 75% attainment threshold. Figure 4.5 shows the efficiency results, where black regions indicate cases where the MOEA required fewer NFE and white indicates the MOEA failed to surpass the attainment threshold. Looking at GD, the majority of the tested MOEAs produced approximation sets with good proximity with 200k or fewer NFE. The few exceptions are NSGA-II, SPEA2, OMOPSO, IBEA and ϵ-NSGA-II on DTLZ3. NSGA-II, SPEA2 and ϵ-NSGA-II also struggled on higher-dimensional DTLZ1 in terms of efficiency. MOEA/D struggled on UF13 and 8D DTLZ7.

Looking at hypervolume and ϵ_+-indicator, low efficiencies occur on UF6, UF8, UF10-UF13 and higher-dimensional DTLZ problem instances. Comparing these results to Figure 4.4, reduced efficiency corresponds with low attainment probabilities. If the algorithm fails to reliably generate approximation sets surpassing the attainment threshold, they will also be marked with low efficiency. On the scalable DTLZ instances, we observe a rapid loss in efficiency as the problem dimension increases. The Borg MOEA, ϵ-MOEA and MOEA/D are the only MOEAs with high efficiency on the higher-dimensional multi-modal DTLZ1 and DTLZ3 instances.

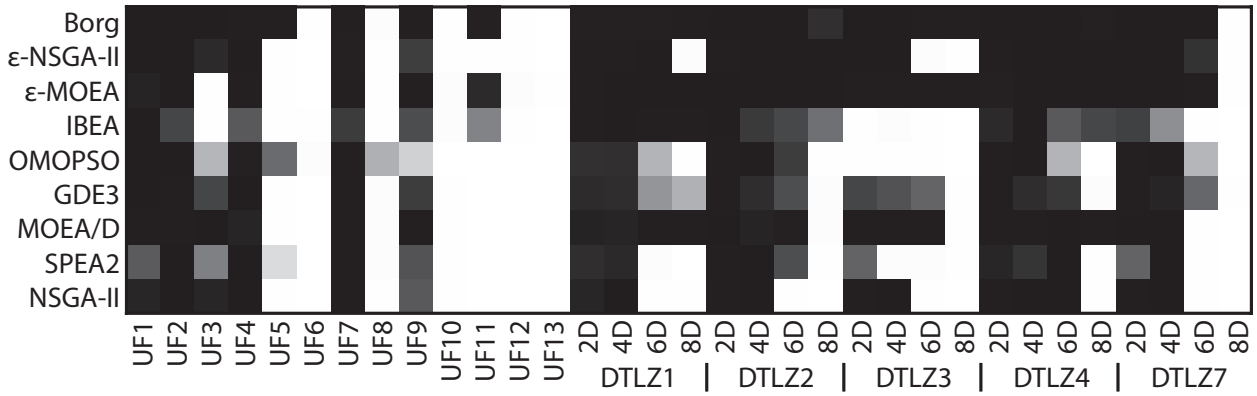
Although the reliability and efficiency of the algorithms are important, it is equally important to understand their controllability. Figure 4.6 shows controllability, which is a measure of the spatial distribution and correlation between parameter sets in the attainment volume. The results are normalized such that the correlation dimensions are divided by the dimension of the hypercube used to sample each algorithm's parameter space. The correlation dimension calculation considers only those parameter sets that are able to attain the 75%-attainment threshold and consequently gives an indication of the distribution of these parameter sets in the full parametric hypervolumes sampled for each algorithm. Cases with low probability of attainment and high controllability signify the attainment volume forms a tightly-clustered sweet spot in a subspace of the overall parameter space. Conversely, cases with high probability of attainment and low controllability indicates the attainment volume is large but sparsely populated.

For example, compare the hypervolume values for the Borg MOEA between Figure 4.4 and Figure 4.6. Figure 4.4 shows that the Borg MOEA has moderate attainment probabilities, but Figure 4.6 indicates the attainment volume is tightly clustered and forms a "sweet spot". IBEA and SPEA2 show the opposite: their higher attainment probabilities correspond often with lower controllability, particularly for GD and ϵ_+-indicator. This suggests these algorithms will be more difficult to parameterize in practice, as the attainment volume is sparse. Overall, the Borg MOEA and ϵ-MOEA are the most controllable of the tested algorithms. They still struggle on several UF problems and 8D DTLZ7. ϵ-NSGA-II and MOEA/D are also strong competitors in terms of GD and ϵ_+-indicator. It is interesting to note that although the Borg MOEA's multioperator search increases its parameterization requirements, its adaptive search actually serves to make the algorithm easier to use and more effective than the other algorithms on most problem instances.

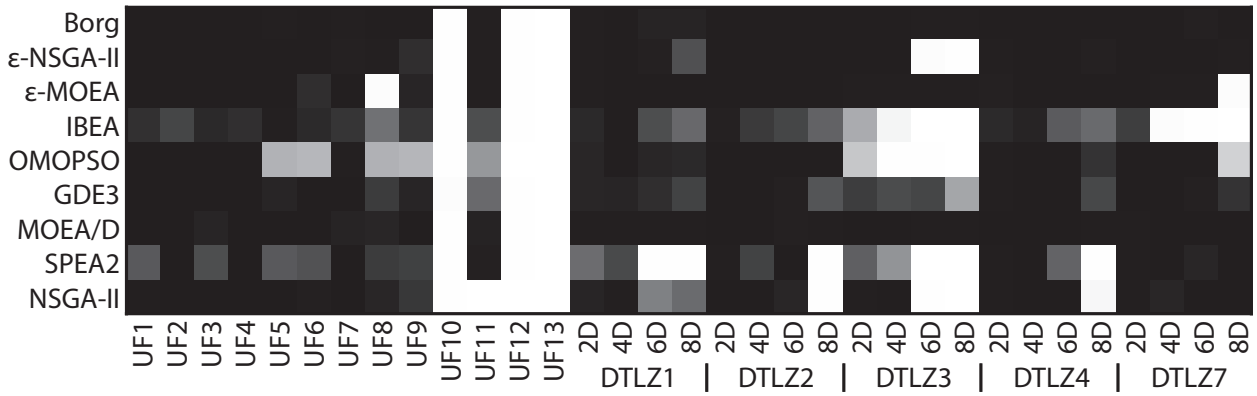
Table 4.3 shows the number of problems each MOEA resulted in the best metric value statistically tied for the best. Ties and statistical differences were determined using a 9-



(a) Generational Distance (GD)



(b) Hypervolume



(c) ϵ_+-Indicator

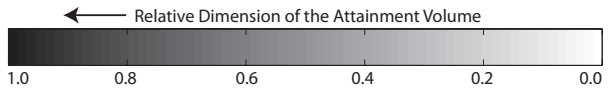


Figure 4.6: Controllability of each algorithm on the problems studied as measured using the correlation dimension. Black regions indicate controllable algorithms with large sweet spots; white regions indicate the algorithm is uncontrollable.

Table 4.3: Statistical comparison of algorithms counting the number of problems in which each MOEA was best or tied for best. The Kruskal-Wallis and Mann-Whitney U tests are used to check for statistical differences in the generational distance, hypervolume and ϵ_+-indicator values across the 50 random seed replicates. Counts are differentiated by the search control metrics: best, probability of attainment (prob), efficiency (eff), and controllability (cont).

	Hypervolume				Generational Distance				ϵ_+-Indicator			
Algorithm	Best	Prob	Eff	Cont	Best	Prob	Eff	Cont	Best	Prob	Eff	Cont
Borg	31	18	17	26	31	27	28	32	30	18	21	28
ϵ-MOEA	23	14	24	24	29	30	30	29	22	22	27	27
ϵ-NSGA-II	19	14	14	19	29	28	27	28	19	18	21	26
OMOPSO	20	15	12	10	29	24	24	25	21	16	17	16
MOEA/D	23	4	19	18	32	24	30	27	27	13	25	27
GDE3	24	8	14	7	32	21	27	23	22	11	20	15
IBEA	18	5	11	7	28	23	25	5	11	5	16	3
NSGA-II	16	8	13	13	26	21	26	25	15	9	19	18
SPEA2	16	3	9	7	26	21	24	10	13	5	13	11

way Kruskal-Wallis test preceding 2-way Mann-Whitney U tests on the results from the 50 random seed replicates using 95% confidence intervals (Sheskin, 2004). These statistical tests help guarantee that any observed differences are not a result of random chance. The MOEAs in Table 4.3 are shown top to bottom in the perceived ordering from best to worst. This ordering is weighted towards the hypervolume metric, as it is the strongest indicator that combines proximity and diversity into a single metric value. Across all performance measures, the Borg MOEA and ϵ-MOEA were superior on the most problems. Borg was most dominant in terms of hypervolume, whereas ϵ-MOEA was dominant on generational distance and ϵ_+-indicator. IBEA, SPEA2 and NSGA-II showed the worst performance among the tested algorithms. The large values seen in Table 4.3 for generational distance indicates that most MOEAs were statistically indifferent from one another with respect to this metric. The wider range of values in hypervolume and ϵ_+-indicator implies a number of MOEAs struggled to produce diverse approximation sets. Overall, algorithms like the Borg MOEA, ϵ-MOEA, ϵ-NSGA-II, OMOPSO and MOEA/D should be preferred in practice. Note that four of these five MOEAs include ϵ-dominance, providing experimental evidence in support of the theoretical findings of Laumanns et al. (2002).

These results combined with the statistical study performed in Hadka and Reed (2012a) helps solidify the dominance of the Borg MOEA over other state-of-the-art MOEAs. The work by Vrugt and Robinson (2007) and Vrugt et al. (2009) focusing on multimethod search supports the observation that while multimethod algorithms increase the number of algorithm parameters, the end result is a more robust and controllable tool. Nevertheless, these results show multimodal and many-objective problems still pose challenges, as is clearly observed when looking at the effectiveness and controllability of algorithms.

Now that a coarse-grained picture of search successes and failures has been established, we now explore a more fine-grained analysis of search controls using global variance decomposition. Figure 4.7 and Figure 4.8 show the first-order and interactive effects of the search parameters for the hypervolume metric for all problems. Each subplot is composed of shaded

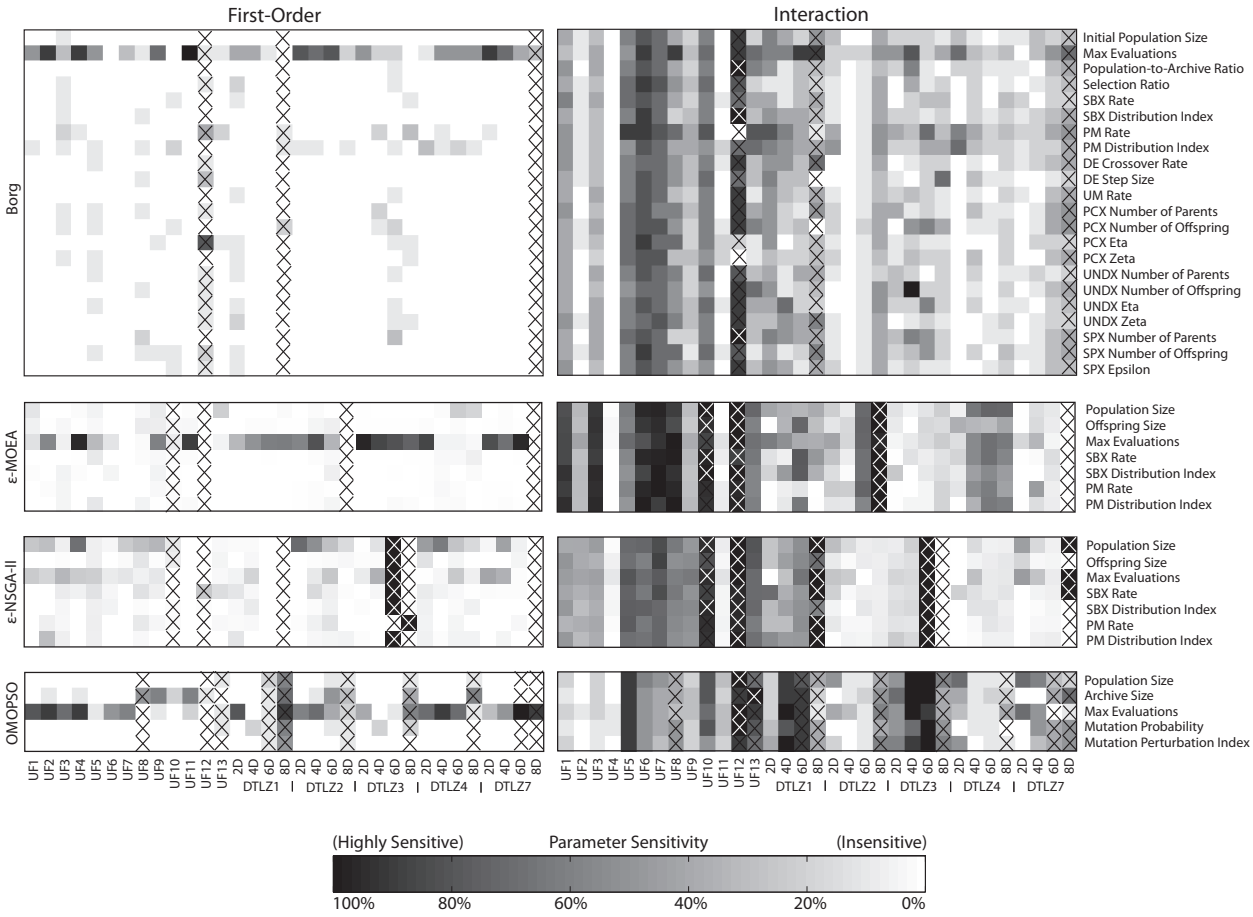


Figure 4.7: Sobol' sensitivities of individual algorithm parameters for all problem instances. The first-order Sobol' indices represent the single parameter contributions to the hypervolume distributions' variances. In a given problem instance, the first order indices for a given algorithm must sum to be less than or equal to 1. Interactive effects represent each parameter's contributions to the hypervolume ensembles variances through combined impacts with other parameters. Note the interactive effects do not sum to 1 for each problem dimension because each shaded cell has variance contributions that are also present in other cells (i.e., higher order interactive parametric effects). X's indicate cases when sensitivities are too uncertain to draw conclusions as determined when the bootstrap confidence intervals exceeded a window greater than +/- 20% around the expected sensitivity value.

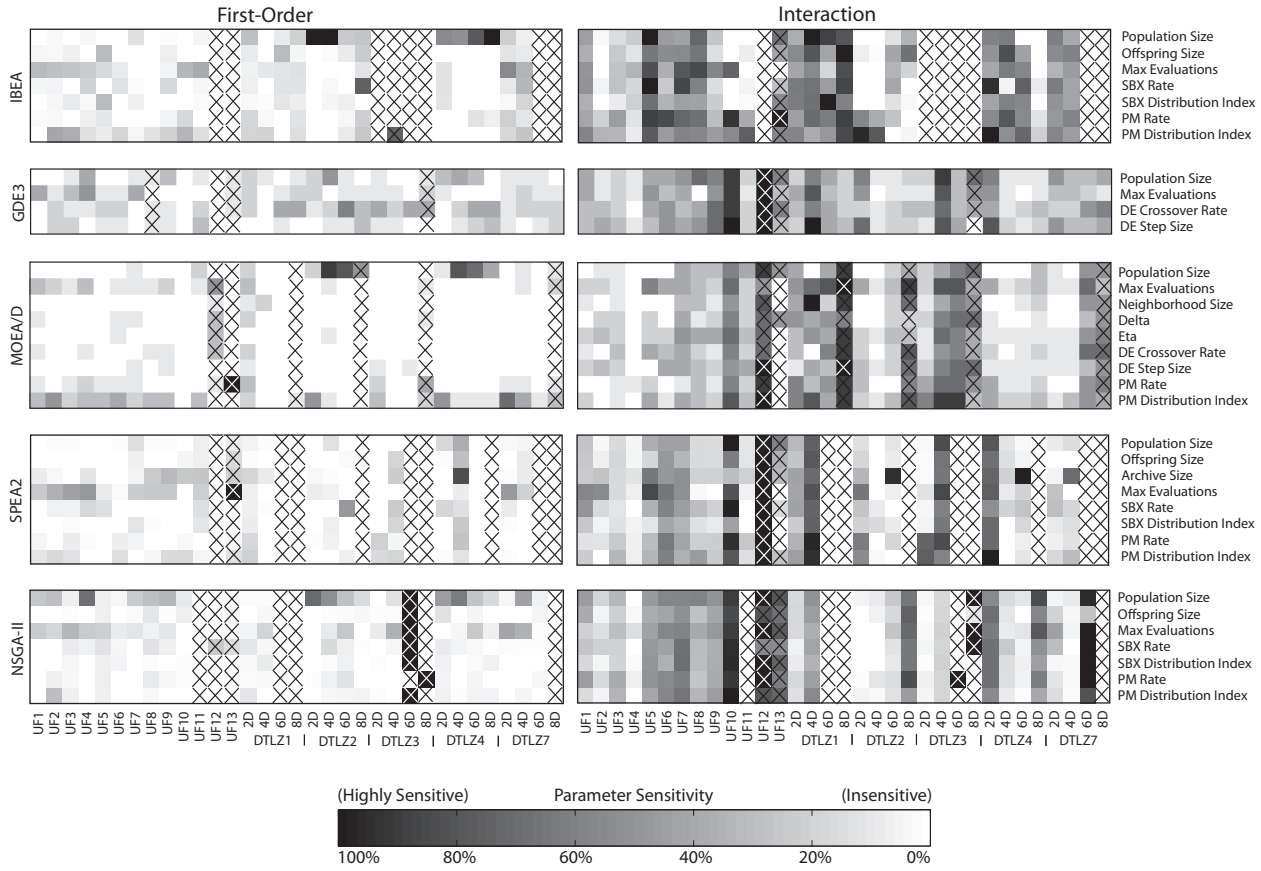


Figure 4.8: Sobol' sensitivities of individual algorithm parameters for all problem instances. The first-order Sobol' indices represent the single parameter contributions to the hypervolume distributions' variances. In a given problem instance, the first order indices for a given algorithm must sum to be less than or equal to 1. Interactive effects represent each parameter's contributions to the hypervolume ensembles variances through combined impacts with other parameters. Note the interactive effects do not sum to 1 for each problem dimension because each shaded cell has variance contributions that are also present in other cells (i.e., higher order interactive parametric effects). X's indicate cases when sensitivities are too uncertain to draw conclusions as determined when the bootstrap confidence intervals exceeded a window greater than +/- 20% around the expected sensitivity value.

squares corresponding to the problem instance (x-axis) and the algorithm's parameters (y-axis). For the DTLZ problems, this visualization captures the change in parameter sensitivities as the objective space's dimension is increased. Black represents the most sensitive parameters whereas white identifies parameters with negligible effects. The shading corresponds to the % ensemble variance contributed by a given parameter or its interactions as identified by Sobol's global variance decomposition. Squares marked with an X indicate the bootstrap confidence intervals exceeded a window greater than +/- 20% around the expected sensitivity value (representing a 40% range), which implies the sensitivity indices could not be reliably computed. A large confidence range in the computed sensitivities is caused by the effects of parameterization not being significantly stronger than stochastic effects (i.e., low signal-to-noise). When this occurs, search is mostly independent of its parameters and is heavily influenced by purely random effects within the evolutionary algorithms. Therefore, we say the X's indicate search failure.

Note Figure 4.7 focuses on the Borg MOEA, ϵ-MOEA, ϵ-NSGA-II and OMOPSO, as these algorithms all share some combination of adaptive operators or ϵ-dominance archives. Figure 4.8 provides the sensitivities for the remaining algorithms. While these figures contains a lot of information, there are several key observations. First, for several problems there are strong first-order effects, indicating one or more parameters are independently responsible for the algorithms' performance. For the Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and OMOPSO, the key first-order parameter across most problems is the maximum number of evaluations. This indicates that parameterizing the Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and OMOPSO should prove easier in practice as the first-order impact of parameters is controlled for the most part by a single parameter, the maximum number of evaluations. Lengthening the runtime of these MOEAs will help produce better results, assuming the optimum has yet to be achieved. As a result, these algorithms should benefit from parallelization, as increasing the number of evaluations should directly result in better performance. Interestingly, these four MOEAs all utilize ϵ-dominance archives, suggesting that ϵ-dominance is an important component for controllability. Table 4.3 and Figure 4.5 also show that the Borg MOEA, ϵ-NSGA-II, ϵ-MOEA and OMOPSO are in fact highly efficient on many problem instances, so it is possible to exploit their sensitivity to NFE to attain effective, reliable and efficient search.

MOEA/D and NSGA-II show strong first-order effects for population size on a number of problems. Hadka and Reed (2012a) show with control maps that these MOEAs require larger population sizes in these cases. As the algorithm runtimes grow polynomially with the population size, MOEA/D and NSGA-II are required to have long runtimes in order to maintain their performance. MOEAs not sensitive to population size will scale better in practice.

Across all tested algorithms we observe a strong trend of increasing interaction effects with increasing objective count. The level of interaction appears dependent on the problem instance, and may reflect problem difficulty. In particular, poor controllability in Figure 4.6 coupled with high levels of interaction between parameters indicate parameterization is difficult for a specific algorithm and problem instance. For instance, on UF11 the Borg MOEA

dominates the other tested algorithms in probability of attainment and controllability, as shown in Figures 4.4 and 4.6. This is reflected in Figure 4.7 in the strong first-order sensitivity to the maximum number of evaluations and weak interactive effects. On the other hand, IBEA and GDE3 show strong first-order and interactive effects spread across multiple parameters. We expect such MOEAs to be difficult to control due to the significance of many parameters. This is confirmed in Figure 4.6 by the weak controllability of IBEA and GDE3 in hypervolume relative to the other tested MOEAs. In this manner, a better understanding of how parameters effect search performance can be deduced from Figures 4.7 and 4.8.

A critical concern highlighted in Figure 4.8 for most MOEAs that do not use ϵ-dominance archives is how their parameter sensitivities change significantly across problem types and even within the same problem with increasing objective dimension. Moreover, their sensitivities have increasingly complex interactive dependencies for many-objective problems. Consequently, a user cannot use any "rule-of-thumb" beyond enumerative testing when using the algorithms in challenging many-objective applications, especially if they are multimodal. These results highlight the importance auto-adaptive search frameworks such as the Borg MOEA that minimize controllability challenges while maintaining efficient and reliable search.

In Hadka and Reed (2012a), we observed that for most problems, only one of the Borg MOEA's recombination operators were probabilistically dominant. In other words, the auto-adaptive multi-operator approach used in the Borg MOEA identified a key operator for each problem. However, Figure 4.7 shows that all of the operators strongly influence the overall hypervolume performance. In Vrugt and Robinson (2007) and Vrugt et al. (2009), the authors observed the same phenomenon in their multimethod approach — while a single operator became probabilistically dominant in search, the remaining operators remained critical to the overall success of the algorithm.

4.3 Conclusion

Due to the increasing interest in using MOEAs to solve many-objective problems, it is necessary to understand the impact of objective scaling on search controls and failure modes. In this study, we contribute a methodology for quantifying the reliability, efficiency and controllability of MOEAs. In addition, this methodology clarifies the multivariate impacts of operator choices and parameterization on search. We have observed that many algorithms have difficulty in maintaining diverse approximation sets on problems with as few as four objectives. In addition, we have shown the necessity of diversity-maintaining archives, such as the ϵ-dominance archive, when applying MOEAs to problems with more than three objectives. A major contribution of this study is our proposed controllability measure, which permits comparing MOEAs without arbitrary parameterization assumptions. Most algorithms are reasonably reliable, efficient and controllable for attaining approximation sets that are in close proximity to the reference sets; however, diversity is far less controllable as a problem's objective space increases in dimension. One of the major factors identified for such search failures is multi-modality and the lack of ϵ-dominance archives.

Sobol's global variance decomposition was used to establish the sensitivities of each algorithm's parameters on the hypervolume of its resulting approximation set. A shift in parameter sensitivities from first-order to interactive effects was observed as the number of objectives is increased. These results can be used by researchers and practitioners when establishing parameterization guidelines. Moreover, these results suggest the need for adaptive search controls for many-objective optimization, while also indicating that adapting search controls will be non-trivial at higher problem dimensions.

The Borg MOEA's multioperator adaptivity strongly enhanced its overall effectiveness, efficiency and controllability relative to the other algorithms tested. The Borg MOEA shows consistent levels of effectiveness, efficiency and controllability for a majority of the problems tested, and had very dominant performance on higher dimensional problem instances. By identifying search control issues, key parameters, and failure modes on test problems, improvements to MOEAs and their potential applicability to real-world problems can be assessed. While this study is only a first step towards understanding the impact of objective scaling on MOEAs, it has yielded several insights into the challenges faced when applying MOEAs to many-objective problems.

Chapter 5

Case Study: Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems

This chapter is drawn from the following paper: "Hadka, D., Simpson, T.W. and Reed, P.M. (2012). *Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems.* IEEE Congress on Evolutionary Computation, Brisbane, Australia, 10-15 June 2012, pp. 986-995."

This chapter explores the application of the Borg MOEA on a real-world product family design problem: the severely constrained, ten objective General Aviation Aircraft (GAA) problem. The GAA problem represents a promising benchmark problem that highlights the importance of using auto-adaptive search to discover how to exploit multiple recombination strategies cooperatively. The auto-adaptive behavior of the Borg MOEA is rigorously compared against its ancestor algorithm, the ϵ-MOEA, by employing global sensitivity analysis across each algorithm's feasible parameter ranges. This study provides the first Sobol' sensitivity analysis to determine the individual and interactive parameter sensitivities of MOEAs on a real-world many-objective problem.

The remainder of this chapter is organized up as follows. Section 5.1 introduces the GAA problem. Section 5.2 presents the experimental details that are based on the diagnostic framework introduced in Chapter 4. Section 5.3 presents the diagnostic assessment results from the GAA problem. Finally, Section 5.4 summarizes the findings of this chapter.

5.1 Introduction

In Chapter 4, we found that the Borg MOEA matched or exceeded the performance of the other MOEAs on the majority of test problem instances, and was particularly effective on the many-objective problems. Reed et al. (2012) extended this study by applying the same nine algorithms to three real-world, water resource engineering applications. Across the three water resource engineering applications, the Borg MOEA again proved to be efficient,

effective, reliable and easy-to-use (i.e., large parameter "sweet spots" (Goldberg, 2002)). This suggests the Borg MOEA is a strong candidate MOEA for application to many-objective engineering design problems.

To further explore the characteristics of the Borg MOEA, this chapter provides a detailed statistical analysis of the Borg MOEA's search controls relative to its non-adaptive ancestor, the ϵ-MOEA, on a severely challenging real-world engineering design problem. It should be noted that our prior comprehensive assessment of MOEAs showed that the Borg MOEA was best overall when compared against eight state-of-the-art MOEAs, and the ϵ-MOEA was a top performer among the non-adaptive traditional MOEAs (Hadka and Reed, 2012b). The algorithms' search controls are rigorously assessed using Sobol' variance-based global sensitivity analysis (Sobol', 2001; Saltelli, 2002; Sobol' and Kucherenko, 2005).

The product family design problem tested in this chapter is the General Aviation Aircraft (GAA) design problem introduced by Simpson et al. (1996). Compared to existing product family design problems (Simpson, 2005), it is a relatively small problem that involves the design of three general aviation aircraft (a product family) that share common subsystems but must satisfy the needs of various general aviation clients. "General aviation" refers to all flights excluding military and commercial operations, and thus caters to a diverse set of potential clientele, from recreational pilots to traveling business executives. As a single aircraft cannot meet all individual needs, three aircraft are designed to accommodate 2, 4 and 6 passengers while satisfying a wide variety of performance and economic constraints. The over-arching goal in the GAA product family design problem is the design of the three aircraft to maximize the commonality of subsystems on all three aircraft to reduce costs while simultaneously addressing the conflicting goal of maximizing the tailored performance characteristics of the individual aircraft.

Simpson et al. (1996) introduced the GAA problem and solved it using a two-objective formulation, but they found that they could not generate feasible points and had to allow 3% constraint violations to attain design alternatives. Subsequent to its introduction, further research into alternate formulations and solution strategies have also struggled when solving the GAA problem (D'Souza and Simpson, 2003; Simpson and D'Souza, 2004). Shah et al. (2011) was the first successful application of an MOEA, the ϵ-NSGA-II (Kollat and Reed, 2006), to the GAA problem, successfully generating a large number of potential constraint-satisfying designs.

To characterize the difficulty of the GAA problem, Shah et al. (2011) performed an experiment where they used Monte Carlo sampling to generate 50 million designs and obtained only four constraint-satisfying designs. Furthermore, these four designs were all dominated by designs produced by ϵ-NSGA-II. This highlights that unguided sampling explorations of the problem hold little to no value for informing the decision-makers. For these reasons, the GAA problem provides a compelling baseline for testing MOEAs on severely constrained problems.

In this chapter, we follow the 27 decision variable, 10 objective and 1 aggregate constraint violation formulation of the GAA problem used by Shah et al. (2011). Table 5.1 lists the decision variables and their allowable range for each aircraft in the family. Table 5.2 lists the

Table 5.1: Design parameters and their respective ranges.

Design Variable	Units	Min	Max
Cruise Speed	Mach	0.24	0.48
Aspect Ratio	-	7	11
Sweep Angle	-	0	6
Propeller Diameter	ft	5.5	5.968
Wing Loading	lb/ft^2	19	25
Engine Activity Factor	-	85	110
Seat Width	inch	14	20
Tail Length/Diameter Ratio	-	3	3.75
Taper Ratio	-	0.46	1

Table 5.2: Objectives and ϵ values.

Objective	Units	Min/Max	ϵ
Takeoff Noise	dB	min	0.15
Empty Weight	lb	min	30
Direct Operating Cost	$/hour	min	6
Ride Roughness	-	min	0.03
Fuel Weight	lb	min	30
Purchase Price	1970 $	min	3000
Flight Range	nm	max	150
Max Lift/Drag Ratio	-	max	0.3
Max Cruise Speed	kts	max	3
Product Family Penalty Function	-	min	0.3

ten objectives being optimized for each individual aircraft. Readers are referred to Simpson et al. (1996) and Shah et al. (2011) for full details on the GAA problem. The ϵ-NSGA-II was not included in this study because our primary focus is demonstrating how the auto-adaptive search operators of the Borg MOEA distinguish its performance from ϵ-MOEA. We have verified that the Borg MOEA is fully superior to the ϵ-NSGA-II on the GAA problem in a separate effort.

5.2 Methodology

Since the GAA problem includes side constraints, it is necessary to extend its design from Chapter 3 to include constraint handling. In this and subsequent chapters, the ϵ-MOEA and the Borg MOEA both employ the constraint handling technique proposed by Srinivas and Deb (1994). Their approach extends binary tournament selection as follows:

1. If both solutions violate constraints, then the one with a lower aggregate constraint violation is selected.
2. If one solution is feasible and the other solution violates constraints, then the feasible solution is selected.
3. If both solutions are feasible, then Pareto dominance is used to select the solution.

Recall that ϵ-MOEA selects one parent from the population and the other from the ϵ-dominance archive. On constrained problems, if no feasible solutions have been found yet, then the ϵ-dominance archive may only contain one solution — the solution that least violates the constraints. This is problematic because the lone solution in the ϵ-dominance archive will always be selected as one of the parents. To avoid this issue, the parent selection mechanism in ϵ-MOEA and the Borg MOEA were modified as follows:

1. If no feasible solutions have been found (i.e., the ϵ-dominance archive contains a single solution), then both parents are selected from the population.
2. Otherwise, if feasible solutions have been found, then select one parent from the population and the other from the archive.

Figure 5.1 shows how constraint handling operates within the multioperator procedure. First, one of the six operators is selected using the operator probability distribution. Second, for an operator requiring k parents, $k-1$ are selected from the population using tournament selection. If the archive contains feasible solutions, then the remaining parent is selected randomly from the archive; otherwise, the remaining parent is selected randomly from the population. Lastly, the resulting offspring are inserted back into the population and archive following the same logic as ϵ-MOEA.

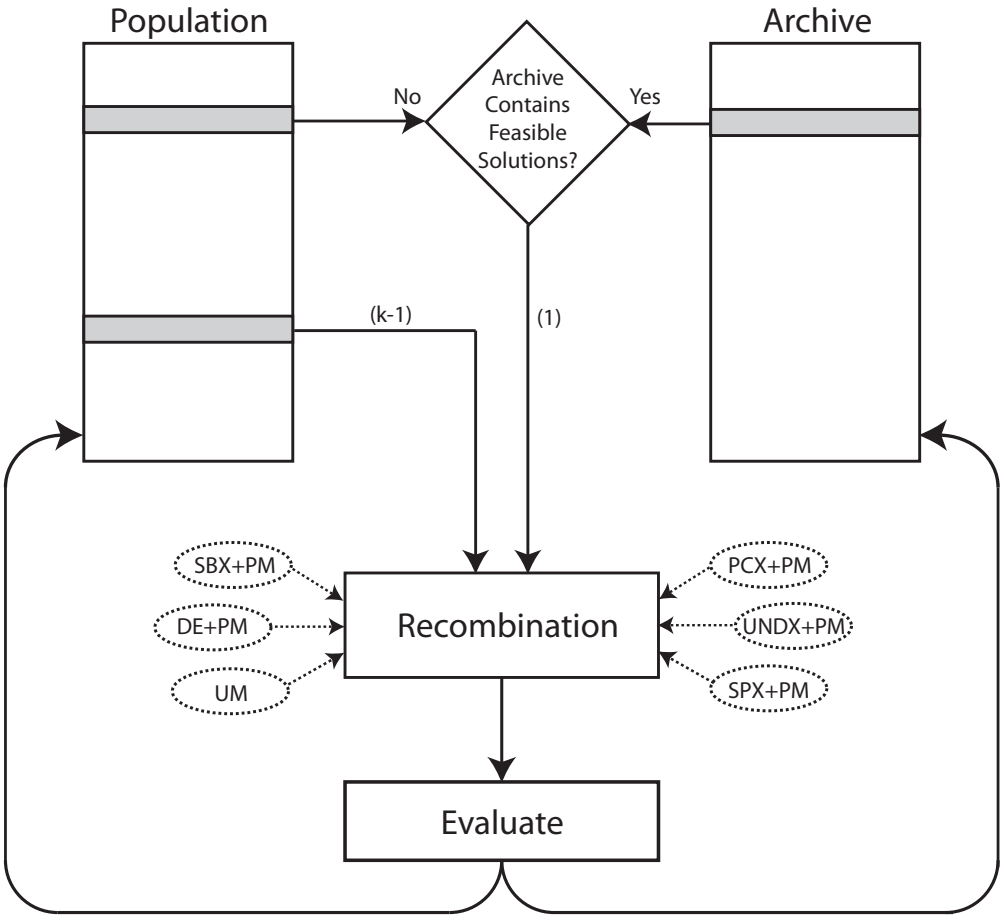


Figure 5.1: Flowchart of the Borg MOEA main loop that includes constraint handling. First, one of the recombination operators is selected using the adaptive multi-operator procedure. For a recombination operator requiring k parents, $k-1$ parents are selected from the population using tournament selection. The remaining parent is selected randomly from the archive if the archive contains feasible solutions; otherwise, it is selected randomly from the population. The offspring resulting from this operator are evaluated and then considered for inclusion in the population and archive.

5.2.1 Sobol' Sensitivity Analysis

As discussed in detail in Chapter 4, Sobol' sensitivity analysis is a form of variance decomposition that attributes the variation observed in a model's output to perturbations of the model's input (Saltelli et al., 2008). In Chapter 4, we explored the application of Sobol' sensitivity analysis to understanding the effects of an MOEA's parameters (e.g., population size, number of function evaluations, mutation and crossover rates) on the end-of-run performance of the algorithm. In this chapter, we extend this analysis by applying Sobol' sensitivity analysis to ϵ-MOEA and the Borg MOEA for the GAA problem.

By using a special pseudo-random sampling technique proposed by Saltelli et al. (2008), one can compute the first-, second- and total-order sensitivity indices using Sobol' sensitivity analysis. For this application, first-order indices reflect the impact of a single input parameter on end-of-run performance, independent of all other parameters. Second-order effects capture the pairwise interactions between parameters, identifying parameter combinations which influence the behavior of MOEAs. Total-order effects sum the first-order effects with all interactive effects (second-order, third-order, and so on) for a given parameter. By capturing these interactions, researchers can identify the parameter combinations that are important to each MOEA.

The results from Chapter 4 show that parameter interactions vary across problems and even vary across the same problem class for different numbers of objectives. When parameter interactions change dramatically across problems, the parameters of an MOEA need to be tuned for each application. It is hypothesized that the auto-adaptive search in the Borg MOEA overcomes these limitations to yield robust search regardless of the parameterization choices (i.e., it has been shown to be highly controllable). The sensitivity analysis in this study attempts to confirm this hypothesis on a real-world problem. Moreover, many existing MOEAs are strongly biased by only considering the directional search provided by the SBX and PM operators. This chapter examines the Borg MOEA's multi-operator dynamics for the GAA product family design problem.

5.2.2 Experimental Setup

To perform Sobol' sensitivity analysis and present robust statistical results in the form of attainment probabilities, each algorithm was run on the GAA problem using parameters sampled across the algorithm's full parameter space (see Table 5.3). The parameter samples are produced using the Sobol' sequence generator, which ensures that the parameters are sampled uniformly from the parameter hyperboxes. For an MOEA with P parameters, the Sobol' sequence generator produces $(2P + 2) * N$ parameterizations. Furthermore, each parameterization is run by the MOEA using 50 random seed replications to fully characterize performance.

This sampling strategy represents a Monte Carlo approximation of each MOEA's full joint probability distribution function (PDF) of performance from which we can rigorously assess the best achieved value and probability of attainment measures. In total, this study accumulates the results of 2000000 sets of MOEA results on the GAA problem.

Table 5.3: Sampled parameter ranges and default settings.

Parameter	Min	Max	Default
(Initial) Population Size	10	1000	100
Max Evaluations	10000	1000000	50000
Injection Rate	0.1	1.0	0.25
SBX Rate	0.0	1.0	1.0
SBX Distribution Index	0.0	500.0	15.0
PM Rate	0.0	1.0	$1/L$
PM Distribution Index	0.0	500.0	20.0
DE Crossover Rate	0.0	1.0	0.1
DE Step Size	0.0	1.0	0.5
UM Rate	0.0	1.0	$1/L$
PCX # of Parents	2.0	10.0	3
PCX # of Offspring	1.0	10.0	2
PCX Eta	0.0	1.0	0.1
PCX Zeta	0.0	1.0	0.1
UNDX # of Parents	2.0	10.0	3
UNDX # of Offspring	1.0	10.0	2
UNDX Eta	0.0	1.0	0.5
UNDX Zeta	0.0	1.0	0.35
SPX # of Parents	2.0	10.0	3
SPX # of Offspring	1.0	10.0	2
SPX Epsilon	0.0	1.0	0.5

Given the computational demands of this study, the codes for ϵ-MOEA and the Borg MOEA were implemented using the MOEA Framework Java library[1] and executed on the CyberSTAR compute cluster at the Pennsylvania State University. CyberSTAR consists of 384 2.66 GHz Intel Nehalem processors and 128 2.7 GHz AMD Shanghai processors.

5.2.3 Performance Metrics

Performance metrics are used to evaluate the approximation sets produced by running an MOEA, allowing the comparison of approximation sets using numeric values. While hypervolume is a preferred performance metric (Fonseca and Fleming, 1996), its use in this study is computationally infeasible due to the GAA problem having 10 objectives. Instead, the following three performance measures are employed by this study, which are detailed in the reference text by Coello Coello et al. (2007).

First, generational distance (GD) is used as a measure of proximity to the reference set. GD is the average distance of approximation set solutions to the nearest reference set solution. Thus, approximation sets nearer to the reference set result in lower GD values.

Second, inverted generational distance (IGD) measures the diversity of the approximation set by averaging the distance of reference set solutions to the nearest approximation set solution. Approximation sets with solutions near each reference set solution yield lower IGD values.

Third, additive ϵ-Indicator (AEI) measures the consistency of the approximation set. Since AEI measures the largest distance ϵ that the approximation set must be translated to dominate the entire reference set, any region of the approximation set that poorly approximates the reference set will result in larger AEI values. An approximation set that consistently approximates the entire reference set will result in lower AEI values.

All three metrics are normalized by the bounds of the reference set. The ideal value of each is 0.

5.2.4 Best, Probability of Attainment and Efficiency

As discussed in Chapter 4, Sobol' sensitivity analysis requires that we globally sample the full parameterization space of each MOEA to approximate the joint PDF for their performance. Consequently, we have defined rigorous measures of their performance to capture (1) the best overall result, (2) the probability of attaining high quality results, and (3) the efficiency in attaining high quality results. These metrics are defined in Chapter 4.

First, the best achieved value records the best metric value achieved across all runs of an MOEA, reflecting the absolute best performance observed using that algorithm.

Second, the probability of attainment records the probability that an MOEA surpasses a threshold of performance. For example, if the threshold is set to 0.1, then the probability of attainment records the number of approximation sets measuring a metric value of ≤ 0.1.

[1] `http://www.moeaframework.org/`

In this experiment, we vary this threshold from 0 to 1 in increments of 0.01, which allows us to show the change in attainment probabilities across a range of performance thresholds.

Finally, efficiency measures the minimum number of objective function evaluations (NFE) required by the MOEA to produce results exceeding a threshold of performance with high probability. This probability is computed by dividing the parameter hyperbox into bands of 10000 NFE each and determining the fraction of parameters in each band that produce results exceeding the threshold. The band with the minimum NFE that attains the threshold with a probability $\geq 90\%$ defines our measure of efficiency. For example, if the threshold is set to 0.1 and an MOEA's efficiency is the band $70000 - 80000$, then running the MOEA on the problem for 80000 NFE has a high likelihood of producing approximation sets measuring a metric value ≤ 0.1 across all of its sampled parameterizations. It is important to generalize performance to this probabilistic context in order to capture efficiency that is robust regardless of how an MOEA is parameterized (i.e., it is efficient and easy-to-use).

5.3 Results

ϵ-MOEA and the Borg MOEA were run using 50 random seed replicates for all of the statistically sampled parameter inputs prescribed in the experimental setup section. Each run produces an approximation set, all of which are combined to form the reference set. This reference set is subsequently used to compute the GD, IGD and AEI metrics. The reference set consists of 630 solutions total, with 112 produced by ϵ-MOEA and 518 solutions produced by the Borg MOEA. Figure 5.2 shows the parallel coordinates plot of the reference set, indicating the solutions produced by ϵ-MOEA and the Borg MOEA. The figure is drawn such that the preferred direction for each objective is toward the bottom of each of the vertical lines for each objective. Figure 5.2 shows that the Borg MOEA finds a far more diverse set of solutions and that its solutions are more effectively discovering the extremes.

Note that we say an algorithm produced the reference set solution if it was not dominated by any other solutions produced by the other algorithm. However, given that both ϵ-MOEA and the Borg MOEA utilize ϵ as a problem-specific resolution for determining significant differences between solutions, we can also ask how many reference set solutions are ϵ-covered by each algorithm. A reference set solution is ϵ-covered by an algorithm if there exists a solution in the approximation set whose distance from the reference set is smaller than ϵ. Percentage-wise, ϵ-MOEA ϵ-covered 38.7% of the reference set when accumulating results across all its runs. The Borg MOEA ϵ-covered 97.3% of the reference set across its runs. This implies the Borg MOEA nearly perfectly specified the entire reference set for this problem. It is also worth noting that the ϵ-values when using ϵ-dominance archiving are not algorithm parameters. They represent the "significant precision" for each objective for an engineering or real world calculation (i.e., in real-world applications, overly precise non-domination is meaningless and can even be harmful). This fact is often lost when studies focus solely on test functions that have no physical meaning.

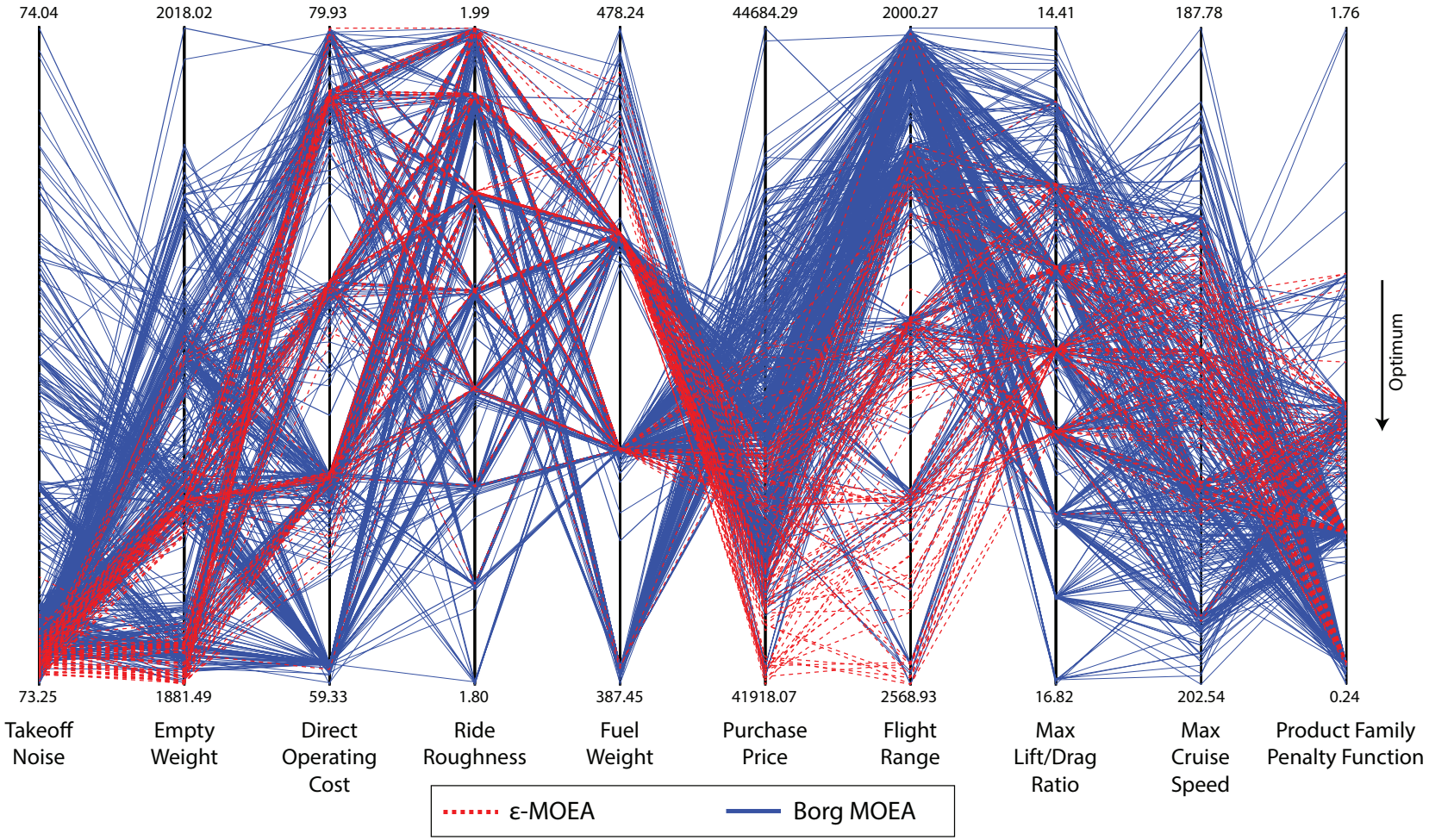


Figure 5.2: Parallel coordinates plot of the reference set generated by ϵ-MOEA and the Borg MOEA. The traces in the plot are colored by the algorithm which produced the solution. The ideal direction for each objective is downwards.

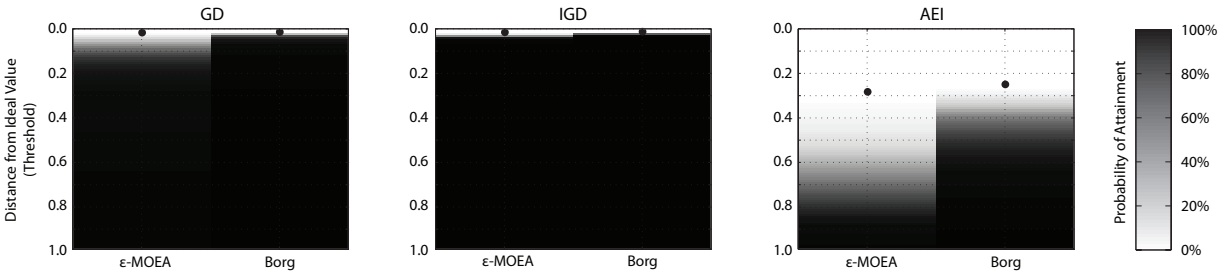


Figure 5.3: Plots showing the best achieved metric value and probability of attainment for each performance metric. The y-axis ranges across the metric values from 0 to 1. The circle markers indicate the best achieved metric value by each algorithm. The shaded bars show the probability of each algorithm producing results which match or exceed a threshold. The threshold is the metric value in the y-axis. Black regions indicate 100% attainment; white regions indicate 0% attainment.

5.3.1 Best Achieved Value, Probability of Attainment and Efficiency

Next, we examine the overall best achieved value and attainment probabilities in Figure 5.3. Each subplot in Figure 5.3 shows the results for the GD, IGD, and AEI metrics for both algorithms. The y-axis provides the distance of the approximation set from the reference set. Ideal performance would have all runs measuring a distance of 0 from the reference set. The solid dots indicate the best achieved metric value for each algorithm (smallest distance) across all of its runs. Thus, a solid dot nearer to the top of each subplot indicates at least one parameterization of the algorithm performed ideally for a given metric. For all three metrics, the Borg MOEA slightly outperforms ϵ-MOEA with regards to the overall best achieved metric value. It should be noted that this is not a very strong measure of performance. Users would be interested being able to attain this level of performance regardless of their parameterization choices (i.e., a high probability of attainment across the sampled parameterizations for each algorithm).

In Figure 5.3, the probability of attainment is shown by the shaded bars. Recall that the y-axis shows the threshold, varying in distance from the reference set. The shading represents the probability of the parameterizations exceeding the threshold value, where black indicates 100% attainment probability and white indicates 0% attainment probability. Intermediate probabilities appear as a shade of gray as noted in the key in the figure. For GD and IGD, the Borg MOEA has a 100% attainment probability up to metric values within a distance of 0.02 of the reference set. ϵ-MOEA, on the other hand, begins to have trouble reaching GD and IGD values within a distance of 0.1 of the reference set.

An even more dramatic difference is seen in AEI. Here, it is very unlikely that ϵ-MOEA produces AEI values less than 0.5. ϵ-MOEA can only reliably attain AEI values larger than 0.8. Recall that AEI is a measure of consistency. This result implies that ϵ-MOEA is very inconsistent, and produces results that poorly approximate some portions of the reference set (i.e., it is prone to gaps in its approximation sets). The Borg MOEA provides more

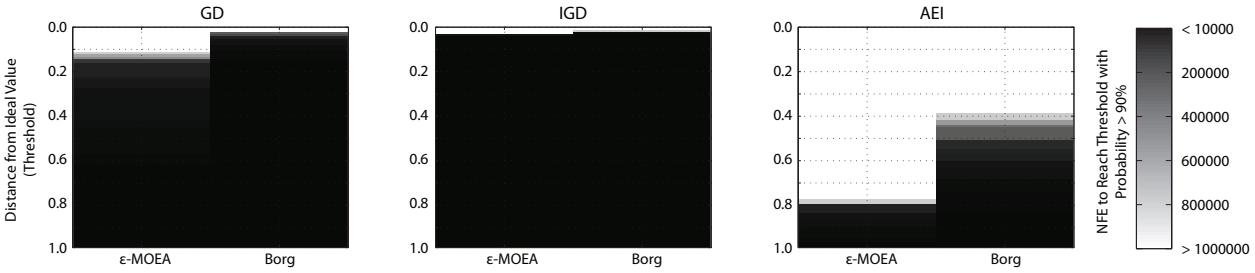


Figure 5.4: Plots showing the efficiency for each performance metric. The y-axis ranges across the metric values from 0 to 1. The shaded bars show the minimum NFE required for each algorithm to match or exceed the threshold of the y-axis. Black regions indicate few NFE are required; white regions indicate more than 1000000 evaluations (the upper limit in this study) are required.

consistent results, showing strong probabilities up to AEI values of 0.25.

Similar in design to Figure 5.3, Figure 5.4 shows the efficiency of the algorithms. Here, the shading indicates the minimum NFE required for the algorithm to produce results meeting or exceeding the threshold of the y-axis with a high probability ($\geq 90\%$ of the parameterizations sampled in a given band of NFE are successful in meeting or exceeding the metric threshold). We observe that the Borg MOEA exhibits substantially higher efficiency than ϵ-MOEA. For GD and IGD, the Borg MOEA can produce results within a distance of 0.05 of the ideal metric value with as few as 50000 NFE. For AEI, the Borg MOEA is dramatically more efficient and effective. ϵ-MOEA requires more than 1000000 NFE to get within a distance of 0.8 of its ideal value. Figure 5.4 in combination with attainment results in Figure 5.3 show that beyond this point, ϵ-MOEA is failing to attain any reliable search across its sampled parameterizations.

5.3.2 Sobol' Sensitivity Analysis

Sobol' sensitivity analysis provides information about the importance of each individual MOEA parameter as well as its interactions with other parameters. Figure 5.5 shows the first-, second-, and total-order sensitivities in each plot with respect to their AEI performance. Around the outside of the plots, the filled circles correspond to each parameter of the algorithms. The size of the circle reflects the first-order sensitivity. A small circle indicates that the parameter has no effect on the performance of the algorithm, whereas a large circle indicates that the parameter has a significant effect on the algorithm. Strong first-order sensitivities are helpful if they exist because they distinguish which parameter(s) users should focus on when using that particular MOEA. The rings around the circles show total-order effects. Total-order sensitivities represent the fully interactive, non-separable multi-parameter controls. Larger rings indicate larger total-order sensitivities. If the rings are significantly larger than the filled first-order circles, then most of a parameter's influence emerges through its interactions with other parameters. The lines between the circles show second-order effects in Figure 5.5. Thicker lines indicate stronger second-order interaction

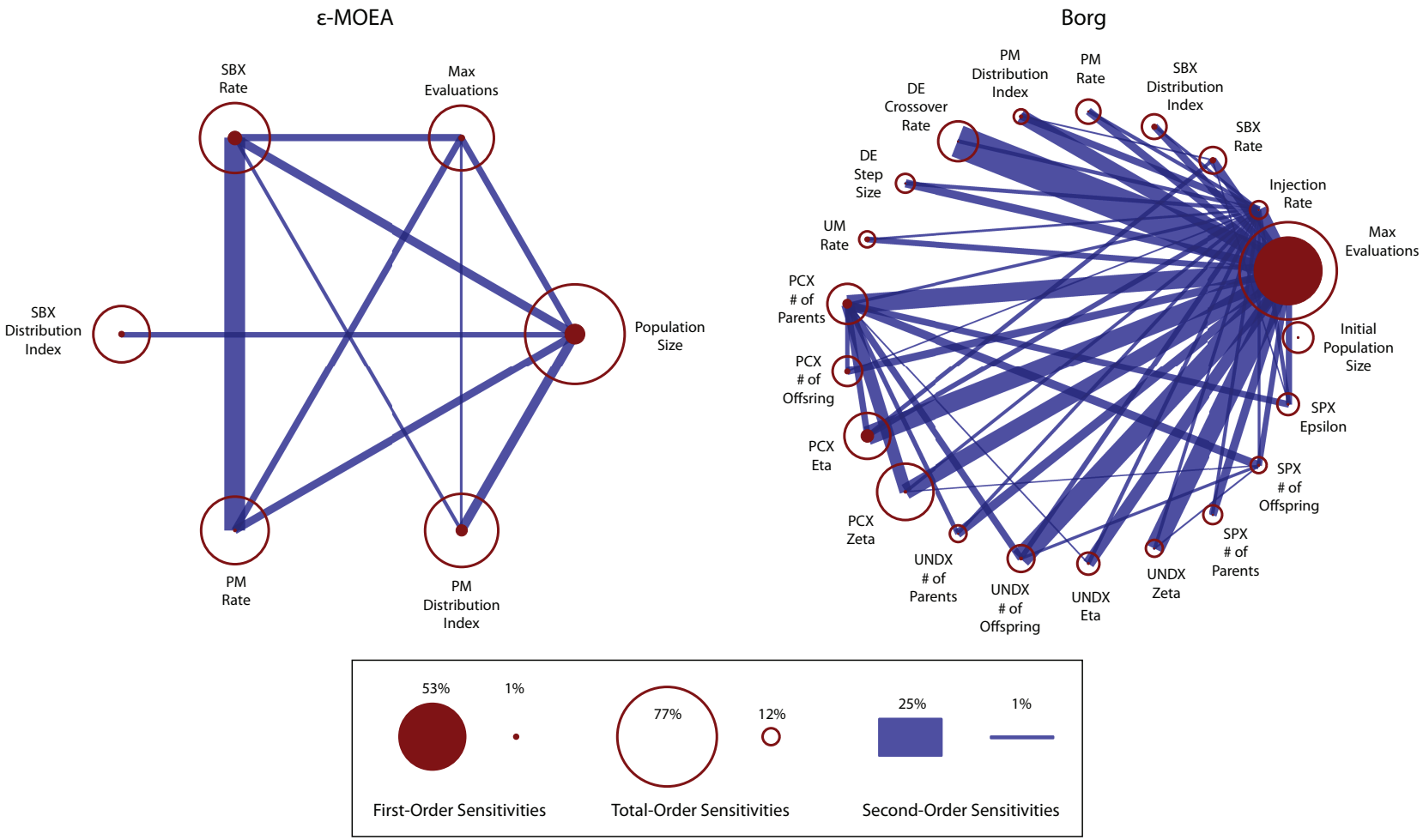


Figure 5.5: First-, second- and total-order sensitivities between the parameters controlling ϵ-MOEA and the Borg MOEA with respect to their AEI performance. The circles represent the first-order sensitivities of each parameter, where larger circles indicate the parameter has a strong impact on performance. Rings around each circle indicate total-order sensitivities, where larger rings indicate the parameter contributes many higher-order interactions. Lines between parameters indicate second-order sensitivities, where thicker lines indicate the two parameters interact strongly to affect performance.

between the two parameters.

Starting with ϵ-MOEA, we see each parameter has small first-order sensitivities, moderate second-order sensitivities, yet large total-order sensitivities. Since the total-order sensitivity for a parameter is a sum of its first-order and all interactive sensitivities involving that parameter, this implies ϵ-MOEA has many higher-order interactions among its parameters. In traditional non-adaptive MOEAs, such as the ϵ-MOEA, strong higher-order interactions among an algorithm's parameters suggest the algorithm is uncontrollable. It will be impossible to independently tune its parameters as they are all fully interdependent. This supports our prior observations that show traditional non-adaptive MOEA's parameter controls are dominantly interactive and change with problem class or dimension of the objective space even within the same problem class (Hadka and Reed, 2012b). This is a severe weakness for real-world application of non-adaptive MOEAs.

The Borg MOEA, on the other hand, shows a strong dependence on the maximum number of objective function evaluations. This does not imply it requires more function evaluations than ϵ-MOEA; alternatively, it means that increasing the number of function evaluations is a clear and easy way to improve the Borg MOEA's performance. This result confirms previous observations of Hadka and Reed (2012b) on this real-world problem. This dependence is shown in the strong first-order sensitivities as well as the strong second-order interactions with other parameters. The remaining parameters have far less effect, showing only small amounts of first-, second- and total-order sensitivities. This suggests that the Borg MOEA is dramatically less sensitive to the parameterization of its operators. Figure 5.5 clearly shows that all of the Borg MOEA's search operators influence its overall performance given their strong interactions with the maximum number of evaluations. It is interesting to note that PCX, SBX and SPX do have some pairwise interactions, which indicate that the Borg MOEA's overall performance is influenced by how these operators work cooperatively.

5.3.3 Auto-Adaptive Operator Probabilities

The Borg MOEA's auto-adaptive and cooperative multi-operator search can be further explored by analyzing the dynamics of its operator probabilities. Figure 5.6 shows traces from 50 seeds of the Borg MOEA using its default parameter settings (shown in Table 5.3). The y-axis of each trace shows the probability each specific operator is used during a run of the Borg MOEA. Figure 5.6 shows that for the first 20000 NFE, three operators are cooperating: SBX, PCX and SPX. Each operator is allocated approximately 30% by the auto-adaptive operator selection mechanism during this initial search phase. This initial search phase accounts for the rapid convergence to the reference set, as observed in the efficiency results (see Figure 5.4). After 20000 evaluations, PCX dominates. PCX's parent-centric behavior is well-suited for introducing small, beneficial perturbations to a design, allowing fine-tuning near the end of a run. Additionally, the strong influence from PCX can be observed in the sensitivity results in Figure 5.5, where PCX shows moderate first- and second-order sensitivities.

Combined with the results presented earlier, Figure 5.6 provides the first evidence of the beneficial effect of multiple search operators on a real-world problem. ϵ-MOEA is limited

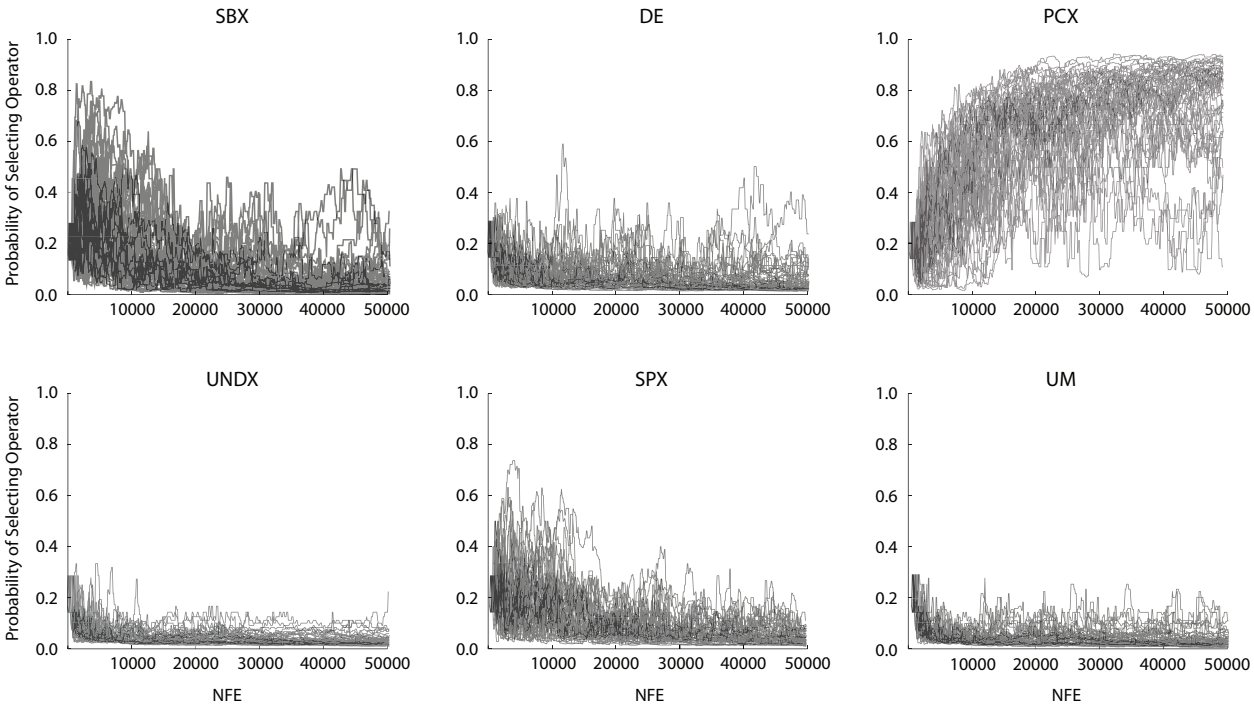


Figure 5.6: Demonstration of the Borg MOEA's auto-adaptive and cooperative multi-operator search, showing the operator probabilities from 50 seeds of the Borg MOEA using its default parameter settings (shown in Table 5.3).

to the SBX operator, but Figure 5.6 shows that other search operators are more effective on this problem. Furthermore, no single operator was sufficient. It is both the individual contributions from one or more operators and their interactions that lead to the behavior seen in the Borg MOEA. Figure 5.5 shows that even non-dominant operators like DE, UNDX and UM have second-order interactions with NFE that influence the Borg MOEA's final performance across its parameterizations.

5.4 Conclusion

In this study, we characterized the effects of the enhancements introduced in the Borg MOEA over its predecessor, the ϵ-MOEA, on the GAA product family design problem. This study also provides the first full Sobol' diagnostic assessment of the Borg MOEA on a severely challenging, real-world, 10-objective test problem. The results show that the enhancements proposed by the Borg MOEA significantly improve reference set coverage and increase the probability of producing high-quality results. Such gains are critical in real-world scenarios, since the decision-maker can be confident that the Borg MOEA is producing high-quality solutions with a high probability with minimal sensitivities to its parameters.

Our results confirm the sensitivities first identified by Hadka and Reed (2012b) on a number of analytical test problems. The Borg MOEA's performance is highly efficient and

can be easily improved by increasing its runtime (i.e., the number of objective function evaluations). This implies two important conclusions. First, by not relying heavily on other parameters, the Borg MOEA is very controllable. The user need not be concerned with parameterization and must only allocate a sufficient amount of processing time in order to produce high-quality results. Second, its dependency on runtime suggests that the Borg MOEA will benefit greatly from parallelization strategies. A simple master-slave architecture will increase the number of objective function evaluations available to the algorithm and, consequently, will improve the quality and reliability of the results. Although the Borg MOEA is dependent on runtime, the efficiency measure demonstrates that significantly fewer NFE are required relative to ϵ-MOEA to produce near-optimal results with high likelihood. This confirms prior work in showing that the Borg MOEA is highly efficient in attaining high-quality Pareto approximation sets in a limited number of evaluations.

Finally, we observed that combinations of operators were active in the Borg MOEA throughout its search. This confirms observations by Vrugt and Robinson (2007) and Vrugt et al. (2009) that multiple operators benefit multiobjective optimization. While we identified SBX, PCX and SPX as the dominant search operators for the Borg MOEA, it is important to note that this does not necessarily imply that the remaining three operators are unnecessary. While DE, UNDX and UM were not selected with high probability, they do contribute to the result by periodically producing new solutions as represented in these operator's second-order interactions with run duration. Overall this study demonstrates that the Borg MOEA is highly controllable in challenging real-world applications and can, consequently, dramatically increase the size and scope of problems that can be effectively addressed. Future work entails tackling other challenging product family design problems and computationally-intensive engineering design challenges encountered in complex systems design.

Chapter 6

Large-Scale Parallelization of the Borg MOEA

This chapter is drawn from the paper "Hadka, D., Reed, P.M., and Madduri, K. *Large-Scale Parallelization of the Borg MOEA for Addressing the Design of Complex Engineered Systems.* Evolutionary Computation, In Review."

This chapter describes two alternative parallel implementations of the Borg MOEA: (1) the master-slave Borg MOEA and (2) the multi-master Borg MOEA. The master-slave implementation is designed to scale up to several thousand processors while the multi-master implementation is designed to scale on emerging Petascale systems (i.e., architectures with greater than 100000 processors). Both implementations retain the auto-adaptive nature of the serial Borg MOEA, but include several improvements to increase the efficiency and reliability of the Borg MOEA on large, complex engineering problems.

The remainder of this chapter is organized as follows. Section 6.1 discusses the need for a large-scale parallel version of the Borg MOEA capable of scaling on emerging Petascale systems. Section 6.2 overviews the serial implementation of the Borg MOEA and discusses several extensions to improve the efficiency and reliability of the algorithm on severely constrained, complex engineered systems. Section 6.3 introduces the master-slave and multi-master implementations of the Borg MOEA. Finally, Section 6.4 concludes this chapter. Analysis of the parallel implementations is provided in Chapter 7 and Chapter 8.

6.1 Introduction

As high performance computing capabilities continue their exponential growth, engineers continue to increase the fidelity of their models and the complexity of the systems considered in design optimization (Venkataraman and Haftka, 2004; Simpson and Martins, 2011; Bloebaum and McGowan, 2010). In these complex engineered systems, decision-makers no longer want a single, numerical solution to a design problem — they instead want the ability to explore and visualize tradeoffs between multiple conflicting objectives to aid them in understanding the range of potential solutions that are available (Balling et al., 1999; Kollat

and Reed, 2007; Simpson and Martins, 2011; Bloebaum and McGowan, 2010).

One of the first design challenges encountered when optimizing complex engineered systems is the formulation of the problem (Woodruff et al., 2013). The problem formulation is often tied to the optimization strategy, and is a potential source for negative decision biases. *Cognitive myopia* refers to the decision bias introduced in highly aggregated, low dimensional formulations for design objectives where optimization may cause stakeholders to inadvertently ignore alternatives that would otherwise influence their decision preferences (Hogarth, 1981). *Cognitive hysteresis* is another form of decision bias where highly restrictive definitions of optimality often reinforce the decision-maker's preconceptions that limit the diversity of alternatives discovered in the process of design (Gettys and Fisher, 1979). The use of highly aggregated objective formulations and severe constraints when abstracting design preferences for a complex engineered system leads to the identification of alternatives in extreme regions of the decision space whose performance is often considered inferior when decision-makers are able to consider additional design relevant performance measures (Franssen, 2005; Brill et al., 1990). Instead, Woodruff et al. (2013) proposes a many-objective visual analytics (MOVA) framework wherein problem formulation, many-objective optimization, negotiated design selection, and interactive visualization are exercised as one fluid process, allowing the problem formulation to change and adapt given new knowledge gained from optimization and visualization. With this approach, the optimization algorithm is given the opportunity to explore higher-dimensional spaces to discover tradeoffs, and the potential for negative decision biases that may result from restrictive definitions of optimality. The benefits of using many-objective formulations has been demonstrated in a number of successful applications (Fleming et al., 2005; Ferringer et al., 2009; Kasprzyk et al., 2009; Reed et al., 2012; Fu et al., 2012; Kasprzyk et al., 2012; Woodruff et al., 2013).

To facilitate the MOVA framework, it is necessary to employ many-objective search tools that can effectively search the higher-dimensional problem formulations while minimizing the time required to explore alternative candidate problem formulations. Rapid evaluations of multiple competing formulation hypotheses allow fluid feedback between the optimization process, design selection, interactive visualization, and problem reformulation. There exist a number of challenges in this regard, particularly when solving complex engineered systems. Complex engineered systems often feature challenging problem properties, including (1) many-objective formulations, (2) multi-modality (or false optima), (3) nonlinearity, (4) discreteness, (5) severe constraints, (6) stochastic objectives, and (7) non-separability (also called epistasis) (Reed et al., 2012). These properties prohibit the use of traditional optimization techniques, and often require the use of metaheuristics like multiobjective evolutionary algorithms (MOEAs). MOEAs are continuing to grow in popularity in many engineering fields as a result of their ability to approximate the set of Pareto optimal (Pareto efficient) solutions in a single run. However, the existence of these challenging problem properties or even subsets of these properties can cause significant search failures in MOEAs (Coello Coello et al., 2007; Nicklow et al., 2010; Wang et al., 2011; Reed et al., 2012; Hadka et al., 2012; Woodruff et al., 2013). For instance, it is well documented that many-objective, multimodal, and non-separable problems can cause many MOEAs to struggle or even outright

fail to find high-quality solutions to an optimization problem (Purshouse and Fleming, 2003, 2007; Ishibuchi et al., 2008a,b; Hadka and Reed, 2012b).

For these reasons, an MOEA that can effectively optimize complex engineered systems within the MOVA framework is desirable. The Borg MOEA is a strong candidate since rigorous studies have demonstrated its ability to solve problems expressing these challenging problem properties, often outperforming other contemporary MOEAs (Reed et al., 2012; Hadka and Reed, 2012a; Hadka et al., 2012). Its strength comes from its auto-adaptive features that tailor the algorithm to specific problem properties. For instance, most contemporary MOEAs employ a single search operator set (i.e., recombination and mutation). The Borg MOEA instead includes a collection of search operators and automatically adapts itself to favor the search operator(s) that are most effective. Often, dynamically varying combinations of search operators yield the best performance. Moreover, prior studies have shown that the Borg MOEA is highly controllable and insensitive to its parameterization (Woodruff et al., 2012; Hadka and Reed, 2012a,b; Hadka et al., 2012; Reed et al., 2012). This provides an important advantage within the MOVA framework: the MOVA decision-maker can focus on their problem versus the design and parameterization of the many-objective search tool. With the Borg MOEA, the algorithm contains a variety of search operators that will remain effective across a range of problem domains (Reed et al., 2012).

There are two remaining challenges before a many-objective search tool like the Borg MOEA can be deployed operationally within the MOVA framework. First, the search tool must support rapid feedback for computationally demanding optimization tasks. Not only are the function evaluation times considered in design applications increasing, but the number of function evaluations (NFE) required to converge to high-quality solutions for highly challenging many-objective applications is also increasing. For example, a problem with an objective function evaluation time of 0.1 seconds running on an MOEA for 1000000 NFE will require 27 hours to complete a single random seed trial, not including any additional overhead required by the MOEA. Second, the search tool must converge to high-quality approximations with high reliability. Since MOEAs are stochastic optimization tools, every run of an MOEA may produce substantially different results. This issue often dramatically increases the demands for highly challenging applications where multiple seed trials are required to compensate for diversity maintenance and convergence failures within the available wall clock for an application (Reed et al., 2012). In the example above, a typical 50 random seed trial analysis increases the computational time for attaining an improved approximation to the Pareto optimal set to more than 50 days. Additionally, if the output of the search tool is not reliable, the decision-maker can not depend on it to provide guidance within MOVA.

This study contributes two parallel variants of the Borg MOEA designed to facilitate rapid and reliable optimization. A goal for this study is to provide reduced wall clock optimization by parallelizing objective function evaluations across thousands or even tens of thousands of processors. This study explores the relative effectiveness, efficiency, and reliability of two alternative parallelization schemes for the Borg MOEA. A simple master-slave Borg MOEA implementation distributes function evaluations across many slave nodes without changing the core serial behavior of the algorithm. Our second parallelization scheme is the

multi-master Borg MOEA implementation, which runs many concurrent instances of the master-slave Borg MOEA. However, unlike the classic island-based model, the multi-master Borg MOEA introduces a global controller node responsible for detecting search failures in individual islands and providing guidance derived from a global archive tracking high-quality solutions and search operators. The multi-master Borg MOEA implementation generalizes the serial algorithm's auto-adaptive search to massively parallel computing architectures, and exploits the enhanced diversity maintenance of the island model for parallelization (Coello Coello et al., 2007; Tang et al., 2007) while simultaneously maximizing the speedup attained with the master-slave scheme.

6.2 The Serial Borg MOEA

The Borg MOEA consists of three key components: (1) an ϵ-dominance archive to maintain a diverse set of Pareto approximate solutions, (2) an ϵ-progress restart mechanism triggered by search stagnation to avoid preconvergence to local optima, and (3) the use of multiple search operators that adapt to a given problem's landscape (Hadka and Reed, 2012a). These components are adaptive in nature, allowing the Borg MOEA to adapt to local search conditions encountered in stochastic, complex engineered systems. Additional complexities of complex engineered systems have necessitated several extensions to the Borg MOEA. This includes (1) constraint handing, (2) improved operator probabilities to handle severely-constrained search, and (3) a new adaptive extension to the ϵ-progress restart mechanism. This section details the serial Borg MOEA with the addition of these new extensions.

6.2.1 Constraint Handling

In Chapter 5, we extended the Borg MOEA to support constraint handling for use with the General Aviation Aircraft (GAA) problem using the technique proposed by Srinivas and Deb (1994). We reiterate this constraint handling mechanism here since its use in severely constrained complex engineered systems is vital. This mechanism extends binary tournament selection as follows:

1. If both solutions violate constraints, then the one with a lower aggregate constraint violation is selected.

2. If one solution is feasible and the other solution violates constraints, then the feasible solution is selected.

3. If both solutions are feasible, then Pareto dominance is used to select the solution.

The Borg MOEA selects one parent from the population and the other from the ϵ-dominance archive as illustrated in Figure 6.1. On constrained problems, if no feasible solutions have been found yet, then the ϵ-dominance archive may only contain one solution — the solution that least violates the constraints. This is problematic because the lone

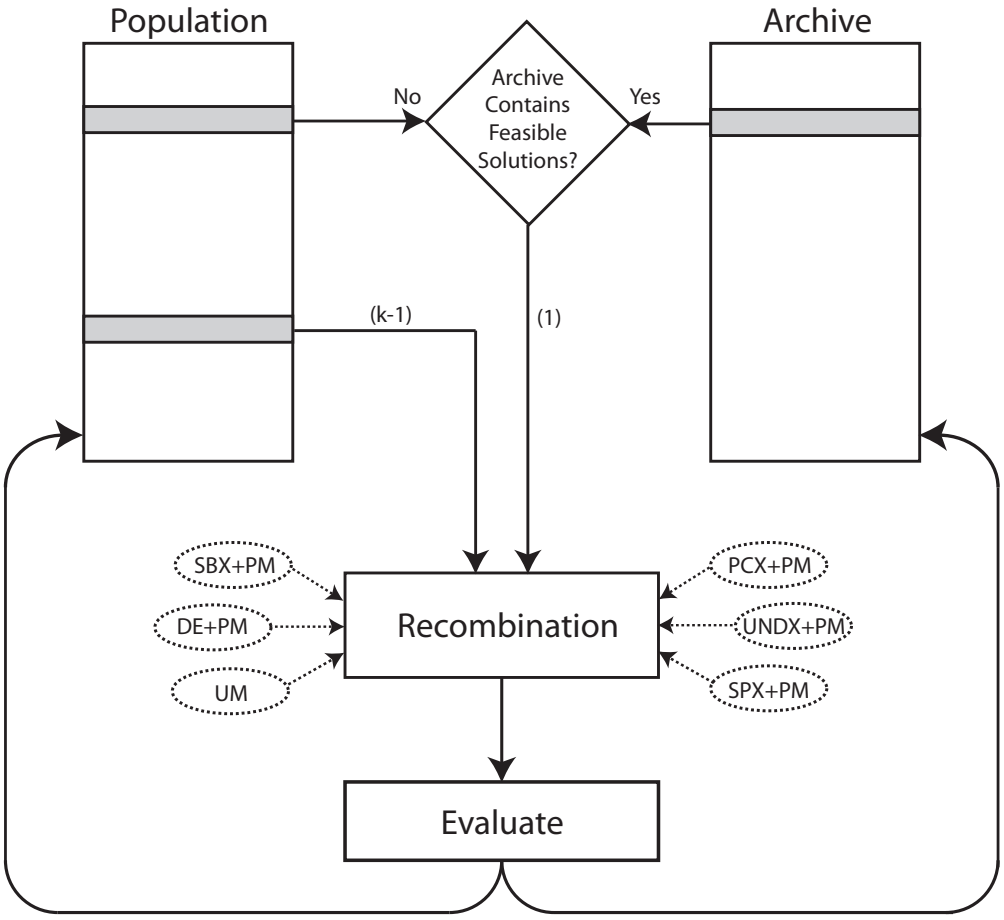


Figure 6.1: Flowchart of the Borg MOEA main loop. First, one of the recombination operators is selected using the adaptive multi-operator procedure described in Section 6.2.2. For a recombination operator requiring k parents, $k-1$ parents are selected from the population using tournament selection. The remaining parent is selected randomly from the archive if the archive contains feasible solutions; otherwise it is selected randomly from the population. The offspring resulting from this operator are evaluated and then considered for inclusion in the population and archive.

solution in the ϵ-dominance archive will always be selected as one of the parents. To avoid this issue, the parent selection mechanism in the Borg MOEA is modified as follows:

1. If no feasible solutions have been found (i.e., the ϵ-dominance archive contains a single solution), then both parents are selected from the population.
2. Otherwise, if feasible solutions have been found, then select one parent from the population and the other from the archive.

Figure 6.1 shows how constraint handling operates within the multioperator procedure. First, one of the six operators is selected using the operator probability distribution. Second, for an operator requiring k parents, $k-1$ are selected from the population using tournament selection. If the archive contains feasible solutions, then the remaining parent is selected randomly from the archive; otherwise, the remaining parent is selected randomly from the population. Lastly, the resulting offspring are inserted back into the population and archive following the same logic as the original Borg MOEA (Hadka and Reed, 2012a).

6.2.2 Auto-Adaptive Multi-Operator Search

One of the problems encountered when using MOEAs is the inability to know *a priori* which recombination operator performs best on a given problem. Moral et al. (2006) first proposed the use of multiple search algorithms in their switching algorithm. This switching approach involves switching randomly to a different search algorithm if certain criteria for progress are not met. Vrugt and Robinson (2007) and Vrugt et al. (2009) proposed a more adaptive approach, AMALGAM, whereby the application of each search algorithm is controlled probabilistically based on the performance attained by each algorithm. The key limitations of both approaches is their use of algorithms that scale poorly on many-objective problems. The Borg MOEA improves upon these designs by integrating multiple search operators in a highly adaptive framework. Furthermore, the full suite of variational operators utilized in the Borg MOEA were not considered in prior multi-method approaches. This is important as the Borg MOEA is better described as an MOEA framework that instantiates itself algorithmically based on which mating and mutation operators or operator combinations are most effective while searching a given problem.

The Borg MOEA exploits the following six search operators:

1. Simulated Binary Crossover (SBX) (Deb and Agrawal, 1994)
2. Differential Evolution (DE) (Storn and Price, 1997)
3. Parent-Centric Crossover (PCX) (Deb et al., 2002a)
4. Simplex Crossover (SPX) (Tsutsui et al., 1999)
5. Unimodal Normal Distribution Crossover (UNDX) (Kita et al., 1999)

6. Uniform Mutation (UM) applied with probability $1/L$

In addition, offspring produced by SBX, DE, PCX, SPX, and UNDX are mutated using polynomial mutation (PM) (Deb and Agrawal, 1994). It should be noted that these operators provide a variety of offspring distributions. For instance, SBX, PCX, and PM produce offspring near one of the parents. Such small perturbations helps fine-tune existing designs. SPX and DE result in larger perturbations, allowing the MOEA to translate across large landscapes efficiently. UNDX produces offspring about the centroid of the parents, quickly converging to valleys in the landscape. UM is the most disruptive of the operators, which aids in adding diversity to the population to prevent preconvergence.

Another key difference between these operators is rotational-invariance. In the ideal case, all decision variables are independent and can thus be optimized independently. However, it is common in complex engineered systems to encounter large amounts of interaction (epistasis) between decision variables. SBX and PM are tailored for problems with independent decision variables. PCX, SPX, UNDX, and DE are rotationally-invariant, and will often perform better on non-separable, epistatic problems.

The Borg MOEA uses all six operators, but adapts the probability that each operator is applied based on the success of each operator from prior iterations. The original Borg MOEA implementation based its operator probabilities on which operators contributed the current members of the ϵ-dominance archive. We call this strategy "membership". Membership favors search operators that contribute high-quality and diverse solutions to the ϵ-dominance archive, but this becomes problematic when the ϵ-dominance archive is small, as the probabilities are based on only a small sample of Pareto approximate solutions. There are three potential causes of small archives: (1) a new solution was added to the archive that dominates most or all of the existing members, (2) the ϵ values are too large, or (3) the problem is heavily constrained and is unable to find feasible solutions.

An alternative strategy is to base operator probabilities on which operators contributed the most recent additions to the ϵ-dominance archive. We call this strategy "recency". Consider how recency handles the three cases above. For case (1), recency will base the operator probabilities on the additions that lead up to the most recent dominating solution. For case (2), recency tracks any addition to the archive, even if the change is within an occupied ϵ-box (only one solution can reside in an ϵ-box, and the algorithm favors those nearest to the optimal corner of the ϵ-box). For case (3), recency will favor those operators that produce solutions with fewer constraint violations, since the solution with the fewest constraint violations is always accepted into the archive. In all cases, recency collects valuable information concerning which operators contributed to the archive.

The Borg MOEA is modified to use both strategies. The operator probabilities are based on both membership and recency. This allows the algorithm to derive operator probabilities from a larger and more informative sample. In this study, we track the most recent 50 additions to the archive when calculating recency. Our implementation uses a bounded list, called the recency list, to keep track of the most recent additions to the archive.

More concretely, given $K > 1$ operators, the Borg MOEA maintains the probabilities $\{Q_1, Q_2, \ldots, Q_K\}$, $Q_i \in [0, 1]$, of applying each operator to produce the next offspring.

These probabilities are initially set to $Q_i = 1/K$ and are updated periodically. To update the probabilities, first count the archive membership contributed by each operator, $\{M_1, M_2, \ldots, M_K\}$, where M_i is the number of solutions in the ϵ-dominance archive that were produced by the i-th operator. Then count recency, $\{R_1, R_2, \ldots, R_K\}$, where R_i is the number of solutions in the recency list that were produced by the i-th operator. Afterward, each Q_i is updated by

$$Q_i = \frac{M_i + R_i + \varsigma}{\sum_{j=1}^{K}(M_j + R_j + \varsigma)}. \tag{6.1}$$

The constant $\varsigma > 0$ prevents the operator probabilities from reaching 0, thus ensuring no operators are "lost" during the execution of the algorithm. In this study, we use $\varsigma = 1$.

6.2.3 ϵ-Progress Triggered Restarts

Since the ϵ-dominance archive is the set of all non-dominated solutions produced by the MOEA, Hadka and Reed (2012a) propose monitoring the ϵ-dominance archive to detect search stagnation. If no new non-dominated solutions are accepted into the ϵ-dominance archive over a period of time, then the MOEA has stagnated. For instance, the MOEA may be stuck at a local optima. This mechanism of monitoring the ϵ-dominance archive for search stagnation is called ϵ-progress. In the Borg MOEA, if the entire population is evolved and the ϵ-dominance archive remains unchanged (no ϵ-progress), then a restart is triggered.

A restart involves several steps designed to help the algorithm escape local optima and introduce additional diversity into the search population. First, the population is emptied. Second, the population is resized relative to the ϵ-dominance archive. Several studies theoretically and experimentally demonstrate that maintaining a population size relative to the Pareto approximate set, as inferred by the ϵ-dominance archive size, helps avoid preconvergence (Horn, 1995; Mahfoud, 1995; Kollat and Reed, 2006; Hadka and Reed, 2012a). Finally, the population is filled with all solutions in the ϵ-dominance archive. Any remaining slots in the population are filled with randomly-selected ϵ-dominance archive members that undergo uniform mutation applied with probability $1/L$. This seeding reintroduces previously-discovered non-dominated solutions into the search population but also introduces additional diversity through the mutation operator.

On complex engineered systems, the small perturbations introduced by a mutation probability of $1/L$ may not be sufficient to escape the local optima. Small perturbations also do not help discover other disjoint Pareto optimal regions. However, simply increasing the mutation probability is not straightforward. Larger perturbations are disruptive, and can slow search by introducing many sub-optimal solutions into the population. We propose in this study an adaptive restart strategy that identifies the smallest mutation probability required to escape the local optima.

The Borg MOEA starts with a mutation probability of $1/L$. Whenever a restart occurs that fails to escape the local optima, the mutation probability is increased. When a restart is successful, the mutation probability is decreased. The speed at which the probabilities change is controlled by a parameter called the "mutation index", m_{index}. This index starts

with value 0 and is incremented or decremented when restarts are unsuccessful or successful, respectively. A restart is unsuccessful if there are two back-to-back restarts with no changes to the ϵ-dominance archive (i.e., the ϵ-progress count remains unchanged). The "maximum mutation index", $m_{\max}$, defines the maximum value of m_{index}. The minimum value is 0. Then, the uniform mutation rate is calculated by

$$\text{mutation rate} = \left[1 + \frac{(L-1)\, m_{\text{index}}}{m_{\max}}\right] / L$$

where L is the number of decision variables defined by the MOP. Hence, when m_{index} is 0, the mutation rate is $1/L$; when m_{index} is equal to $m_{\max}$, the mutation rate is 100%.

6.2.4 Controllability of the Borg MOEA

We conclude this section by discussing briefly the results of Chapters 3 and 4. It is commonly known that MOEAs are often strongly sensitive to their parameterizations (Purshouse and Fleming, 2003, 2007). Most contemporary MOEAs are flawed in this respect since their performance is tied to non-trivial parameterizations that are not consistent across problem domains (or even problems within the same domain). Hadka and Reed (2012b) developed a rigorous statistical framework for assessing the sensitivity of MOEAs to their parameterization. MOEAs with highly-sensitive parameters are termed "uncontrollable", as the decision-maker is required to constantly tweak parameters to improve performance. Controllability is a fundamental requirement for MOEAs to have operational value. Our studies have shown for a wide variety of problems that traditional non-adaptive MOEAs often suffer from isolated islands of effective parameters that would be very difficult if not impossible to discover in a real world application context (Hadka et al., 2012; Reed et al., 2012; Woodruff et al., 2012). Moreover, the transition to massively parallel computing systems often limits the amount of compute time available to users, making it of paramount importance that an MOEA lack sensitivity to its parameterizations.

Chapter 4 used this sensitivity analysis framework to rigorously confirm that the auto-adaptive features of the Borg MOEA drastically improves the algorithm's controllability. Several of our studies using test functions and real-world applications have confirmed that while the Borg MOEA typically meets or exceeds other MOEAs in is efficiency, NFE is the key controlling parameter (Hadka and Reed, 2012b; Hadka et al., 2012; Woodruff et al., 2012; Reed et al., 2012). Furthermore, since NFE is its key controlling parameter, it is expected to benefit substantially from parallelization.

6.3 Parallelizing the Borg MOEA

This section describes two parallel implementations of the Borg MOEA. Both implementations are designed to remain faithful to the adaptive nature of the serial Borg MOEA described in Section 6.2. The master-slave Borg MOEA implementation in Section 6.3.1

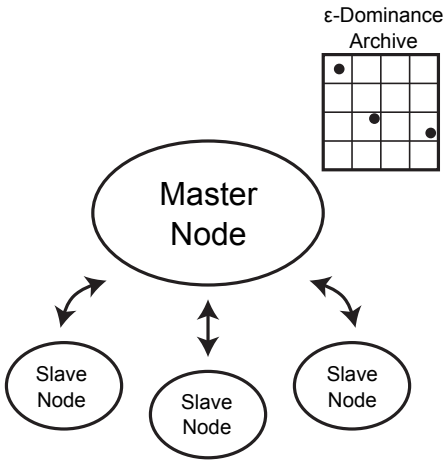


Figure 6.2: Diagram of the master-slave implementation of the Borg MOEA. The master node maintains the ϵ-dominance archive and runs the main loop of the serial Borg MOEA. The decision variables are transmitted to the slave nodes, and the evaluated objective function values and constraints are returned to the master node.

is designed to scale to thousands of processors. The multi-master implementation in Section 6.3.2 expands on the master-slave implementation to scale on emerging Petascale high-performance computing architectures.

6.3.1 Master-Slave Implementation

The master-slave model for MOEAs is a straightforward extension of a serial MOEA to perform objective function evaluations in parallel (Cantú-Paz, 2000; Coello Coello et al., 2007). Modern parallel systems are typically comprised of many multi-core processors, each consisting of two or more processing cores (e.g., a quad-core processor). Throughout this dissertation, we refer to these individual processing cores as "processors". As shown in Figure 6.2, on a system with P processors, one of the processors is labeled the "master" and the remaining $P - 1$ processors are labeled "slaves". Internally, the master node runs the serial MOEA as-is; the only alteration is that objective function evaluations are dispatched to one of the available slave nodes. The master sends the decision variables to an available slave node, the slave node evaluates the problem with the given decision variables, and when finished sends the evaluated objective values and constraints (if any) back to the master node.

Most MOEAs in use today are generational, meaning that the population is evolved in distinct stages called generations. In a single generation, the population is evolved to produce offspring, the offspring are evaluated, and the offspring are added back into the population (possibly replacing existing members in the population). Parallelizing a generational MOEA using the master-slave approach is fairly straightforward (Cantú-Paz, 2000; Coello Coello

et al., 2007). For the sake of simplicity, assume that the number of offspring is equal to the number of slave nodes, $P - 1$. Then, when the algorithm reaches the point where it needs to evaluate the offspring, each member of the offspring is sent to its own slave node for evaluation. Once all slave nodes return the evaluated objective values, the algorithm resumes its serial loop. The need to completely evaluate all offspring before continuing to the next generation gives rise to the term "synchronous MOEA".

For completeness, we can remove our assumption that the number of offspring equals the number of slave nodes, $P - 1$, by sending multiple offspring at a time to a single slave node. For instance, given 16 total processors, 15 would be slave nodes. For an offspring population size of 100, we can batch 6 or 7 offspring to be evaluated by a single slave node. When the offspring size does not divide evenly by the number of slave nodes, then some nodes must process an additional offspring. As a result, some nodes have more work than the others, and will require more time to complete their evaluations. This potentially lowers efficiency as some of the slave nodes will sit idle while others continue processing. This problem also arises when the time to evaluate a solution is variable.

Alternatively, the Borg MOEA is a steady-state algorithm. Steady-state algorithms do not have defined generational boundaries; instead, each individual in the population evolves inside its own distinct evolutionary cycle. Since no boundary exists between generations, such algorithms are also called "asynchronous MOEAs". Additionally, the lack of a boundary often reduces overhead and increases the parallel performance of the algorithm. The remainder of this section describes the master-slave Borg MOEA implementation.

The master-slave Borg MOEA maintains a queue of unevaluated solutions. Whenever a slave node is available for processing, it queries this queue for the next solution to evaluate. If the queue is empty, then the typical Borg operator selection and offspring generation mechanism is triggered to insert one or more offspring (unevaluated solutions) into the queue. Otherwise, the next unevaluated solution in the queue is sent to the slave node. The main Borg MOEA loop is this process of slave nodes querying the queue for solutions, and new solutions being generated and appended to the queue as needed.

When a slave node finishes evaluating a solution and sends the evaluated objective and constraint values to the Borg master node, these solutions are immediately added to the population and ϵ-dominance archive. The strategy for adding/replacing solutions in the population and archive are identical to the serial Borg MOEA. These newly-added solutions are now available as parents when the offspring generation mechanism is invoked next. The flowchart of these steps is shown in Figure 6.3. The other components of the Borg MOEA, such as ϵ-progress restarts, adaptive population sizing, etc., occur next and are identical to the serial Borg MOEA. The only difference is that any new solutions generated during a restart are appended to the queue.

Initialization works similarly to the serial Borg MOEA with one exception. As with offspring generation, the solutions generated during initialization are added to the queue and processed as described earlier. However, consider what happens when running on $N + 1$ processors, with 1 master node and N slave nodes, and an initial population size of N. All N initial solutions will be generated randomly and sent to the slave nodes for evaluation.

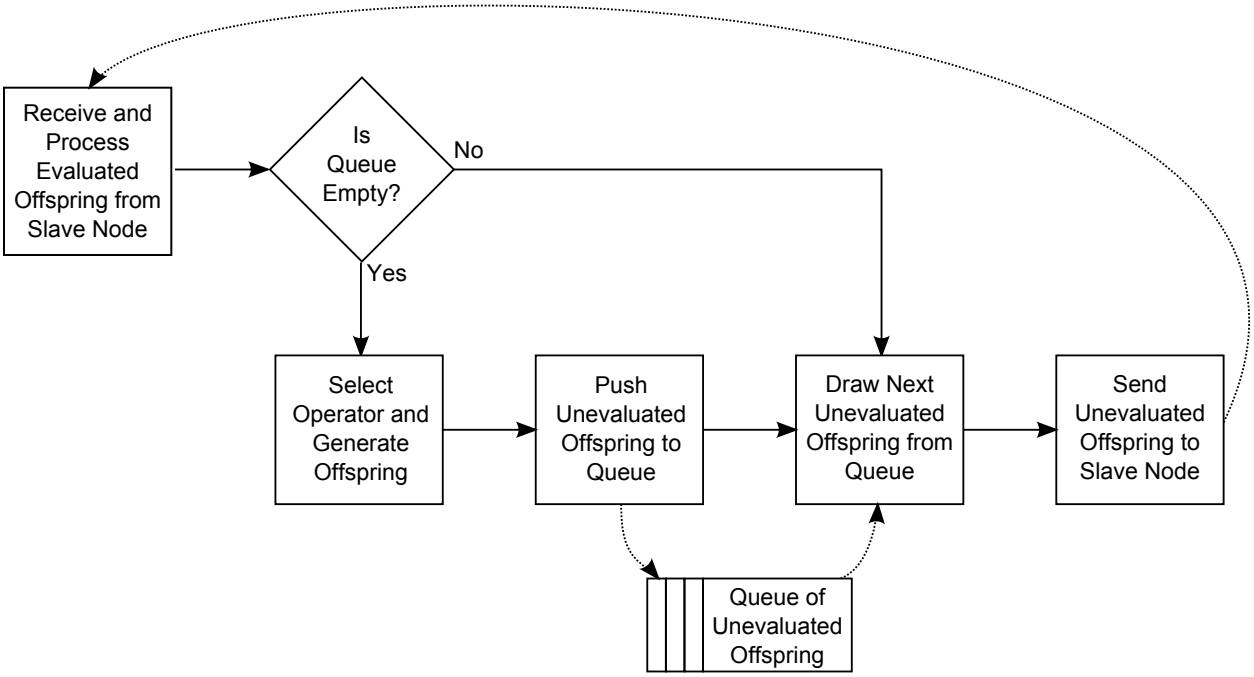


Figure 6.3: Flowchart of the main Borg MOEA loop running on the master nodes. A queue supports the asynchronous generation and evaluation of offspring. When a slave node is available (it returns an evaluated offspring), the master queries the queue for the unevaluated offspring. If the queue is empty, the algorithm invokes the operator selection and offspring generation steps from the serial Borg MOEA.

The first solution to finish evaluation is added to the population, and the next offspring is immediately generated. At this point, the population has only 1 evaluated solution, which is problematic for multi-parent recombination operators and also lacks sufficient genetic diversity. To ensure that the population is filled with a sufficient number of solutions before applying the evolutionary operators to generate offspring, the master-slave Borg MOEA always generates at least $2N$ initial solutions, where N is the number of slave nodes. This ensures that at least N solutions have been added to the population prior to applying any evolutionary operators.

Function evaluation times are a significant factor in controlling the scalability of the master-slave scheme (Cantú-Paz, 2000). As function evaluation times decrease below 1 second, our prior work (Hadka et al., 2013) has shown that the Borg MOEA's master node becomes congested due to rapid objective function evaluation turnaround by the slave nodes, which reduces the efficiency of the algorithm and results in an overall decline in parallel scalability beyond one thousand processors. These results motivated our exploration of the multi-master scheme introduced in this chapter, which represents a hierarchical parallelization scheme (Cantú-Paz, 2000) that hybridizes the classic island-model and master-slave strategies. In the next section, we introduce the multi-master Borg MOEA.

6.3.2 Multi-Master Implementation

The multi-master Borg MOEA abstracts the master-slave implementation to follow an island-based model of parallelization (Cantú-Paz, 2000). In an island-based model, each island runs a distinct MOEA with its own population evolved independently of other islands. Implementations of island-based MOEAs often include periodic migration events, wherein a small fraction of the population at each island is transmitted to one or more other islands. These migrations are intended to permit sharing of information between islands.

While the classic island-based model is a popular strategy for parallelizing MOEAs, it exacerbates the parameterization and algorithmic design challenges present in MOEAs. In order to run an island-based MOEA, one must select (1) the number of islands, (2) the number of processors per island, (3) the population size on each island, (4) operator selection and parameterization, (5) whether to run the same MOEA (homogeneous) or different MOEAs (heterogeneous) on each island, (6) migration policies; etc. Cantú-Paz (2000) developed theoretical models to determine problem-specific values for some of these settings, but in doing so also shows the complexities and non-linear relationships between the various settings that makes parameterization challenging. The effectiveness of an island-based MOEA is heavily dependent on such non-trivial design choices that must be tailored to individual problems. This is a limiting factor in the operational value of classic island-based MOEAs.

Our design of the multi-master Borg MOEA seeks to generalize its ease-of-use and auto-adaptivity while maximizing its parallel efficiency on large-scale computing architectures. Several studies have shown that the Borg MOEA's auto-adaptivity eliminates parameterization concerns by allowing the algorithm to adapt and maximize its potential on a given problem (Hadka and Reed, 2012b; Hadka et al., 2012). This eliminates issues (3), (4), and (5), since the dynamics of the Borg MOEA automatically configure the algorithm for the lo-

cal conditions encountered during search. This additionally implies that each island running the Borg MOEA can assume drastically different search operators as needed to maximize performance. For instance, a struggling island can introduce heavy mutations to escape local optima while another island is fine-tuning near-optimal solutions using small, local perturbations. We address (6) by introducing an auto-adaptive migration mechanism based on the search progress made by each island. Unlike the unguided migration events in classic island-based models, in the multi-master Borg MOEA migrations only occur when an island is struggling and injects high-quality solutions and new search operator preferences to guide the struggling local population. Lastly, we answer (1) and (2) in Chapter 7 by contributing a discrete event simulation model to predict topologies for the multi-master Borg MOEA that maximize its parallel efficiency. The full details of the multi-master Borg MOEA implementation are given below.

As shown in Figure 6.4, the multi-master Borg MOEA introduces a new node, called the "controller", that has two responsibilities: (1) maintaining a global ϵ-dominance archive, and (2) providing guidance to master nodes when they need help. The global ϵ-dominance archive maintains the Pareto optimal solutions discovered by all master nodes. Identical to how each master node uses the ϵ-dominance archive to track the operators that contribute new, Pareto approximate solutions, the controller uses the global ϵ-dominance archive to track the operators that contribute globally Pareto approximate solutions. Note the term "global" as used here refers to the aggregate of all ϵ-dominant solutions from the full suite of searching master nodes. Each master node periodically sends an update to the controller every 10000 NFE. This update contains any new Pareto approximate solutions discovered by the master since its last update.

Since each master node is running an instance of the master-slave Borg MOEA, it includes all of the mechanisms to detect search stagnation and trigger restarts. In the event that these mechanisms are unsuccessful at escaping the local optima, the master node notifies the controller that it needs assistance. Upon receiving the help request, the controller seeds the master with the contents of the global ϵ-dominance archive and global operator probabilities. This in essence replaces the local ϵ-dominance archive that was stuck at a local optima with the global search state. Additionally, it provides the best-known global operator probabilities for contributing new Pareto approximate solutions. Upon receiving this guidance from the controller, the master updates its internal state and triggers a restart. Since the local archive of the master node is now set to the global ϵ-dominance archive, the solutions injected during the restart are derived from the global search state, and the adaptive population sizing ensures the population is resized appropriately given the global search state.

The multi-master implementation also features a different style of initialization from the serial and master-slave Borg MOEA implementations. The original Borg MOEA generated the initial population by sampling the decision variables uniformly at random from their bounds. While uniform sampling is a common initialization strategy used in MOEAs, it has a major disadvantage: it makes no guarantee that the sampled points are representative of the actual distribution. In the context of MOEAs, this means that there is no guarantee that the initial population includes a representative sampling of all decision variable combinations.

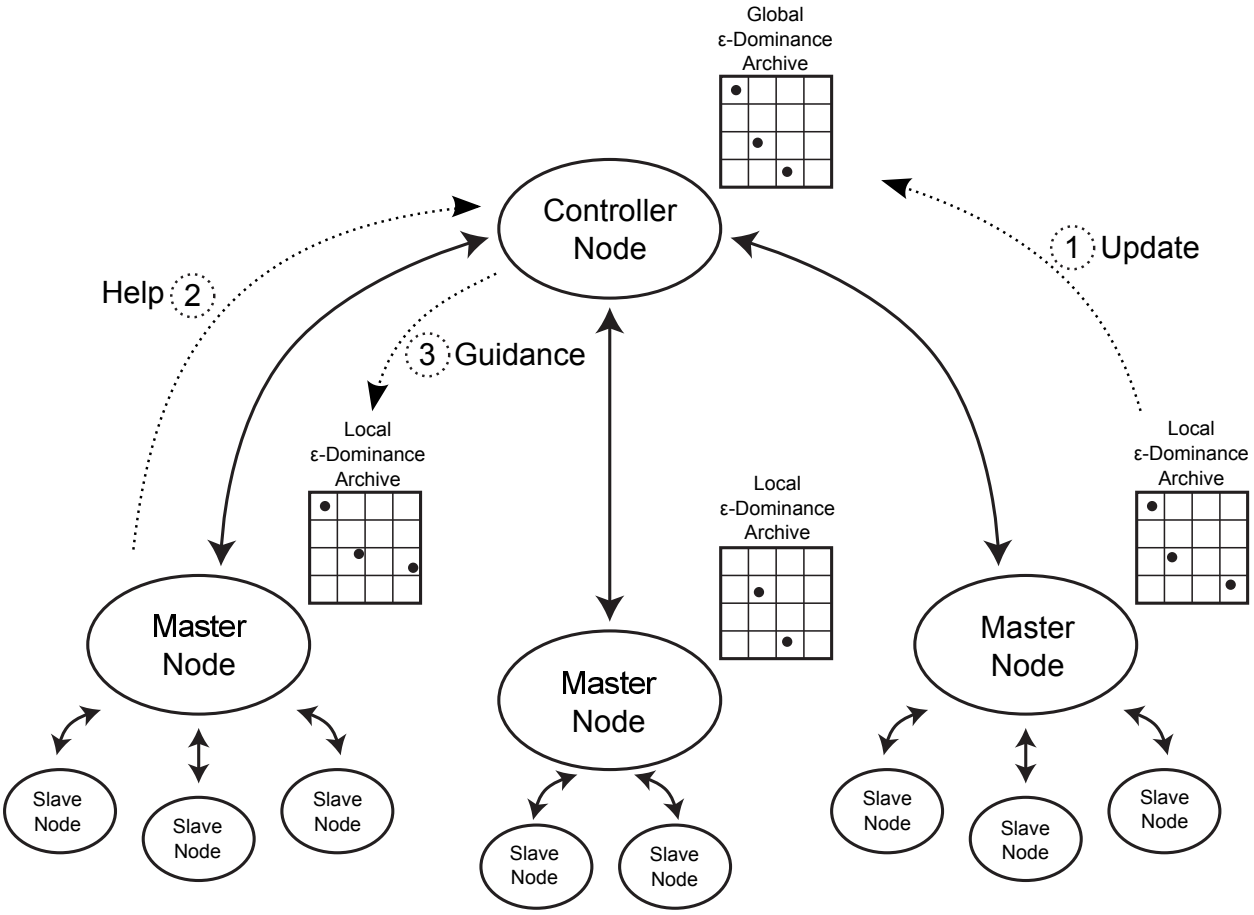


Figure 6.4: Diagram of the multi-master implementation of the Borg MOEA. The multi-master Borg MOEA consists of two or more master-slave instances. This diagram depicts three such instances. The multi-master consists of an additional controller node, which communicates with the masters using several messages. (1) Each master node periodically transmits its local ϵ-dominance archive to the controller to update the global archive. (2) When a master node is struggling, it sends a help message to the controller. (3) The controller responds with guidance, which includes the global ϵ-dominance archive and global operator probabilities.

Instead, uniform sampling tends to result in areas with higher and lower densities, potentially introducing random bias into the initial search population.

It has been proposed in the literature to use other sampling techniques like Latin hypercube sampling (LHS) and Sobol's low-discrepancy sequence generator (Bäck et al., 1997). The improved quality of the samples by LHS and Sobol' sequence have been used in Monte Carlo simulations to improve convergence and reduce the number of required samples (Macdonald, 2009). In the context of MOEAs, LHS and Sobol' sequence help ensure that the initial population contains a representative sampling of the search space. In the multi-master Borg MOEA, we propose the global Latin hypercube initialization strategy. When the multi-master algorithm first starts, each master node notifies the controller of its desired initial population size. The sum total is the number of initial solutions generated by the controller using LHS. The controller then uniformly at random partitions these solutions into the initial populations for each master. Finally, the controller transmits the initial populations to the master.

For example, suppose we have 16 islands each using an initial population size of 100. Just like the master-slave Borg MOEA, the master node generates twice as many initial solutions as required to ensure that the population is filled prior to entering the main evolutionary loop. Thus, each island would request 200 initial solutions. Then, the controller would generate $16 * 200 = 3200$ initial solutions using LHS. Next, these 3200 solutions will then be randomly partitioned into 16 groups of 200. Finally, each group is sent to the corresponding island.

While this initialization strategy adds some additional overhead at startup, it has the benefit of ensuring that globally, the multi-master algorithm starts with a well-distributed, diverse set of initial solutions. Without this approach, the initial populations would have less diversity and likely subject to faster preconvergence.

6.4 Conclusion

This chapter introduced the master-slave and multi-master Borg MOEA. The next two chapters explore the theoretical and experimental properties of these two parallel variants. Chapter 7 explores the theoretical scalability results and provides a strategy for determining the optimal topologies for the parallel Borg MOEA variants. Chapter 8 applies both parallel variants to a real-world complex engineered system: a risk-based water supply portfolio planning problem. This case study exercises the theoretical models on a real-world problem.

Chapter 7

Scalability of the Parallel Borg MOEA

This chapter is drawn from the paper "Hadka, D., Madduri, K. and Reed, P.M. (2013). *Scalability Analysis of the Asynchronous, Master-Slave Borg Multiobjective Evolutionary Algorithm.* International Parallel and Distributed Processing Symposium (IPDPS), Nature-Inspired Distributed Computing Workshop (NIDISC), Boston, MA, 20-24 May 2013 (To Appear)."

This chapter begins with an experimental analysis of the scalability of the master-slave and multi-master Borg MOEA on the Texas Advanced Computing Center (TACC) Ranger supercomputer. Here, we contrast naive speedup (increased NFE) versus hypervolume speedup (increased search quality) resulting from different configurations of the parallel Borg MOEA. Afterward, we build a discrete event simulation model for predicting the runtime behavior of the parallel Borg MOEA. These results are validated using the 5-objective DTLZ2 and UF11 problems. Finally, we discuss how to configure the Borg MOEA to maximize search quality.

The remainder of this chapter is organized as follows. Section 7.1 uses the DTLZ2 and UF11 problems to experimentally explore the scalability of the master-slave and multi-master Borg MOEA. Section 7.2 derives the analytical and discrete event simulation model for predicting the behavior of the parallel Borg MOEA. Section 7.3 discusses how this information is used to configure the master-slave and multi-master Borg MOEA to maximize performance. Finally, Section 7.4 discusses the impact of these results.

7.1 Experimental Scalability Analysis

In Chapters 4 and 5, we identified that the key parameter for controlling the search quality of the Borg MOEA is NFE, the number of objective function evaluations performed in a run. This has been the motivating factor for parallelizing the Borg MOEA, as parallelization should increase NFE and as a consequence search quality. This relies on the assumption that NFE and search quality are correlated. To test this assumption, we must distinguish between speedup that increases NFE, which we term *naive speedup*, and speedup that results in an MOEA attaining high-quality results faster, which we term *hypervolume speedup*. In

this section, we experimentally analyze the naive speedup and hypervolume speedup of the Borg MOEA on a simple 5-objective test problem, DTLZ2, and the more challenging, non-separable UF11 problem.

Naive speedup is speedup in terms of increased NFE as a result of parallelizing the MOEA. Naive speedup is related to parallel efficiency, as a more efficient parallel algorithm yields more NFE. For instance, if two MOEAs are run for the same amount of wallclock time, then the more efficient algorithm will evaluate more solutions. We therefore define efficiency as

$$\text{efficiency} = \frac{\text{NFE}_P}{P \cdot \text{NFE}_S}, \tag{7.1}$$

where NFE_S is the total NFE for a serial MOEA and NFE_P is the total NFE for a parallel MOEA with P processors. The denominator calculates the total NFE that would result from running P instances of the serial MOEA on P processors. The numerator is the actual NFE from running the parallel MOEA on P processors. A parallel MOEA with an efficiency near 1 is ideal.

Hypervolume speedup measures how much quicker a parallel MOEA reaches a certain hypervolume threshold than its serial counterpart. Hypervolume values range from $[0, 1]$ with values near 1 indicating high-quality approximation sets. Hypervolume speedup is calculated by first setting the hypervolume threshold, $H \in [0, 1]$. Then, we determine the minimum wallclock time required for the serial MOEA to attain the hypervolume threshold, T_S^H, and the minimum wallclock time for the parallel MOEA to attain the same threshold, T_P^H. Finally, the hypervolume speedup is calculated by

$$\text{hypervolume speedup} = \frac{T_S^H}{T_P^H}. \tag{7.2}$$

For example, if it takes 100 seconds for the serial MOEA to attain a hypervolume threshold of $H = 0.9$ and only 25 seconds for the parallel MOEA to attain the same threshold, then the hypervolume speedup is 4.

Naive speedup reflects the efficiency of the parallel MOEA and the increase in NFE resulting from lower algorithm overhead. Hypervolume speedup is more indicative of the benefits of parallelization since a parallel MOEA with a larger speedup is attaining high-quality results faster. However, we consider naive speedup in this analysis due to our observation that NFE is the only parameter that effects the search quality of the Borg MOEA. Therefore, we hypothesize that there is a correlation between naive speedup (i.e., parallel efficiency) and hypervolume speedup. To test this hypothesis, we apply the master-slave and multi-master Borg MOEA variants to two well-known analytical problems. The first, the 5-objective DTLZ2 (Deb et al., 2002b), is considered simple for MOEAs to solve (as was shown in the diagnostic analysis in Chapter 4). All decision variables are separable, and can be evolved independently of the others. The second problem, the 5-objective UF11 (Zhang et al., 2009b), is a variant of DTLZ2 where the decision variables are rotated and scaled to introduce dependencies between the variables. Results from the IEEE CEC 2009 competition (Zhang and Suganthan, 2009) and the diagnostic analysis of Chapter 4 show many state-of-the-art

Table 7.1: Notation used throughout this chapter.

Symbol	Description
T_F	Function evaluation time
T_C	Communication time
T_A	Algorithm overhead (population management, offspring generation, etc.)
T_S	Runtime of the serial algorithm
T_P	Runtime of the parallel, asynchronous algorithm
N	Number of function evaluations
P	Number of processors (i.e., cores)

MOEAs struggle to solve UF11. These functions provide a mechanism to explore how problem difficulty interplays with speedup to impact the Borg MOEA's search. Table 7.1 shows the notation used throughout this chapter.

The function evaluation time of these two problems is less than 1 microsecond. To explore the scalability of the Borg MOEA, controlled delays were introduced to the problems. In this chapter, we explore three different delays: 0.001, 0.01, and 0.1 seconds with a coefficient of variation of 0.1. This allows us to precisely control T_F and vary it proportionally to T_C and T_A to measure the actual speedup and efficiency.

For each problem and time delay, the master-slave Borg MOEA was executed on the TACC Ranger system with 16, 32, 64, 128, 256, 512, and 1024 processors. Each run of the algorithm was replicated 50 times, and the reported results are averaged across the 50 replicates. The multi-master Borg MOEA can scale to larger processor counts, and therefore was tested on 1024, 2048, and 4096 processors with 2, 4, 8, 16, and 32 islands.

The experiments performed in this chapter were executed on the Texas Advanced Computing Center (TACC) Ranger system. TACC Ranger consists of 3,936 16-way symmetric multiprocessing (SMP) compute nodes, each containing four 2.3 GHz AMD Opteron Quad-Core 64-bit processors and 32 GBs of memory. Each core can perform 9.2 GFLOPS. In total, there are 62976 processing cores. Throughout this dissertation, we refer to these individual processing cores as "processors". Nodes are connected using two large Sun InfiniBand DataCenter switches.

Figure 7.1 shows the average parallel efficiency of the master-slave Borg MOEA on the 5-objective DTLZ2 and UF11 test problems calculated using (7.1). The three line series plot the different function evaluation times. As one would expect, when T_F is small (i.e., $T_F = 0.001$ seconds), the master-slave Borg MOEA is inefficient as the number of processors grows. For larger T_F, the master-slave Borg MOEA remains efficient at higher processor counts. This observation is consistent across both DTLZ2 and UF11.

Observe in Figure 7.1 that the parallel efficiency is maximized with 16 processors when $T_F = 0.001$, 128 processors when $T_F = 0.01$, and 512 processors with $T_F = 0.1$ seconds. At larger processor counts than listed, the parallel efficiency of the master-slave Borg MOEA begins to decline. It is at this point where NFE is maximized, and we want to determine if hypervolume speedup is also maximized.

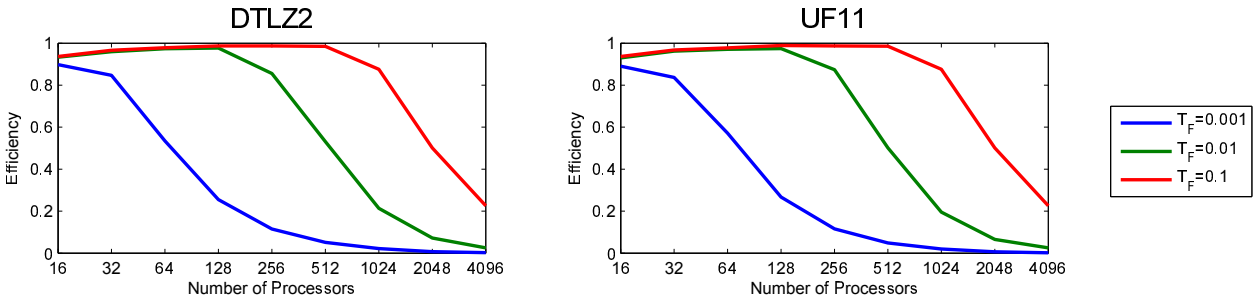


Figure 7.1: The average parallel efficiency of the master-slave Borg MOEA on the 5-objective DTLZ2 and UF11 test problems.

Figure 7.2 shows the average hypervolume speedup of the master-slave Borg MOEA on the 5-objective DTLZ2 and UF11 test problems calculated using (7.2). Here, the hypervolume threshold is set on the x-axis, and the hypervolume speedup for the given hypervolume threshold is plotted on the y-axis. The various line series plot the different processor counts. In these plots, hypervolume speedup is computed relative to the base 16 processor configuration. For this reason, the 16 processor line is flat. A hypervolume speedup of 4 indicates that configuration achieved the same hypervolume as the 16 processor configuration in only 1/4 the wallclock time.

Observe in Figure 7.2 for $T_F = 0.001$ that the 32 processor configuration exhibits the largest hypervolume speedup. This is close to the prediction based on parallel efficiency, which expected the 16 processor configuration to maximize hypervolume speedup. Note that all the larger processor counts have lower hypervolume speedup than the baseline 16 processor configuration for $T_F = 0.001$. The communication overhead quickly dominates the runtime, decreasing the efficiency of the algorithm when T_F is small. As T_F increases to 0.01 seconds, the 128 processor configuration maximizes hypervolume speedup. This matches the prediction based on efficiency. With $T_F = 0.1$ seconds, the 512 and 1024 processor configurations offer maximum hypervolume speedup, closely matching the prediction based on efficiency. This confirms our hypothesis that maximizing parallel efficiency is closely tied to maximizing hypervolume speedup for the master-slave Borg MOEA.

Figure 7.2 also shows how hypervolume speedup is affected by problem difficulty. As the hypervolume threshold is increased to 1, it is more challenging for the MOEA to generate approximation sets surpassing that threshold. This is exhibited by the non-linear curves in Figure 7.2. This is very pronounced in the UF11 subplots, which show a decline in hypervolume speedup as the level of quality increases from a hypervolume of 0 to 0.5, but then an increase in hypervolume speedup as the quality increases further towards 1. This demonstrates that problem difficulty can have a significant impact on search quality and affect the efficacy of parallel MOEAs.

Switching now to the multi-master Borg MOEA, Figure 7.3 shows the average parallel efficiency of the multi-master Borg MOEA on the 5-objective DTLZ2 and UF11 test problems. Here, efficiency is plotted against the number of islands in the multi-master topology. The various line series plot different processor counts. Each subplot in Figure 7.3 shows

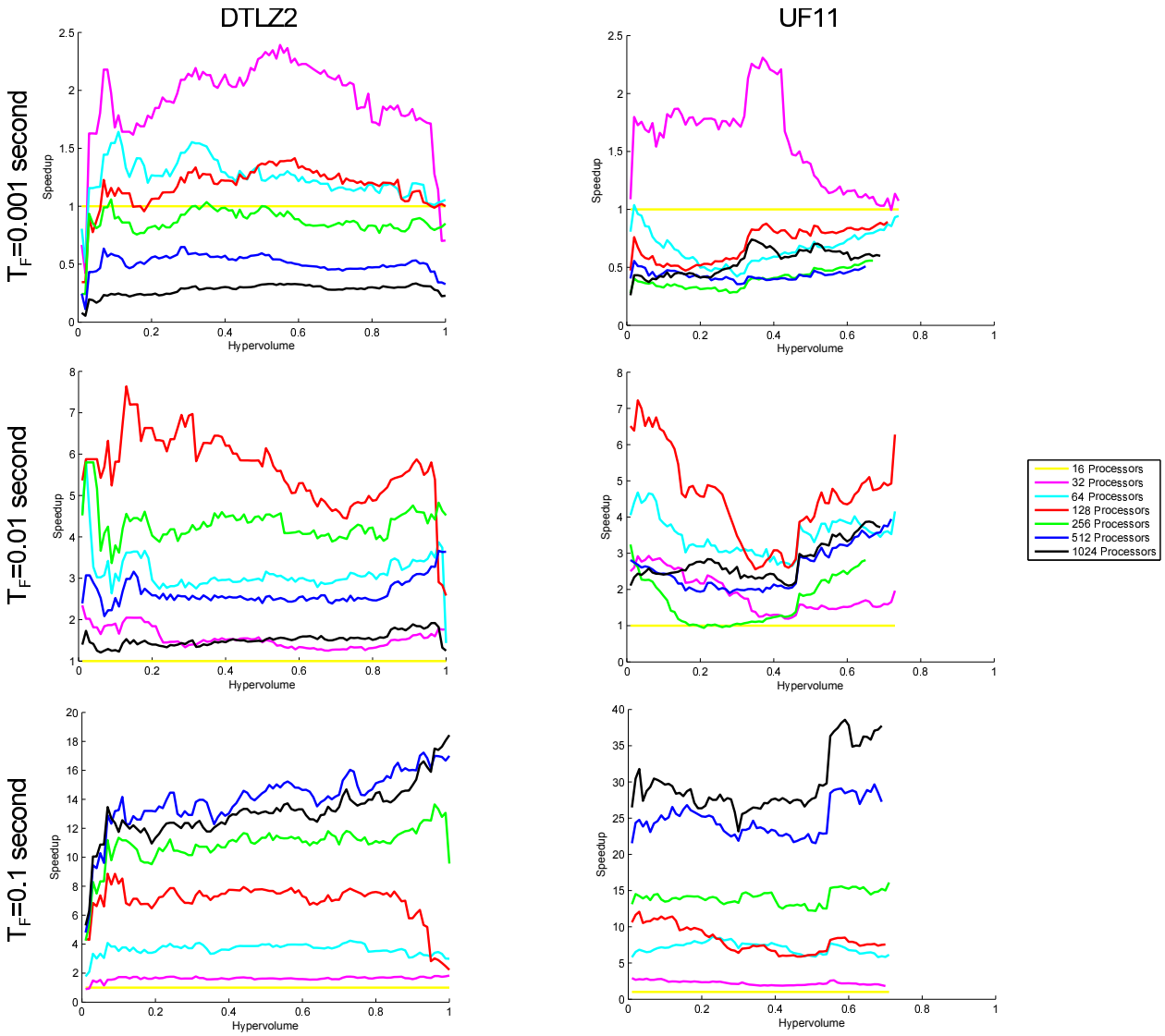


Figure 7.2: The average hypervolume speedup of the master-slave Borg MOEA on the 5-objective DTLZ2 and UF11 test problems. The 16 processor configuration is used as the baseline for calculating hypervolume speedup.

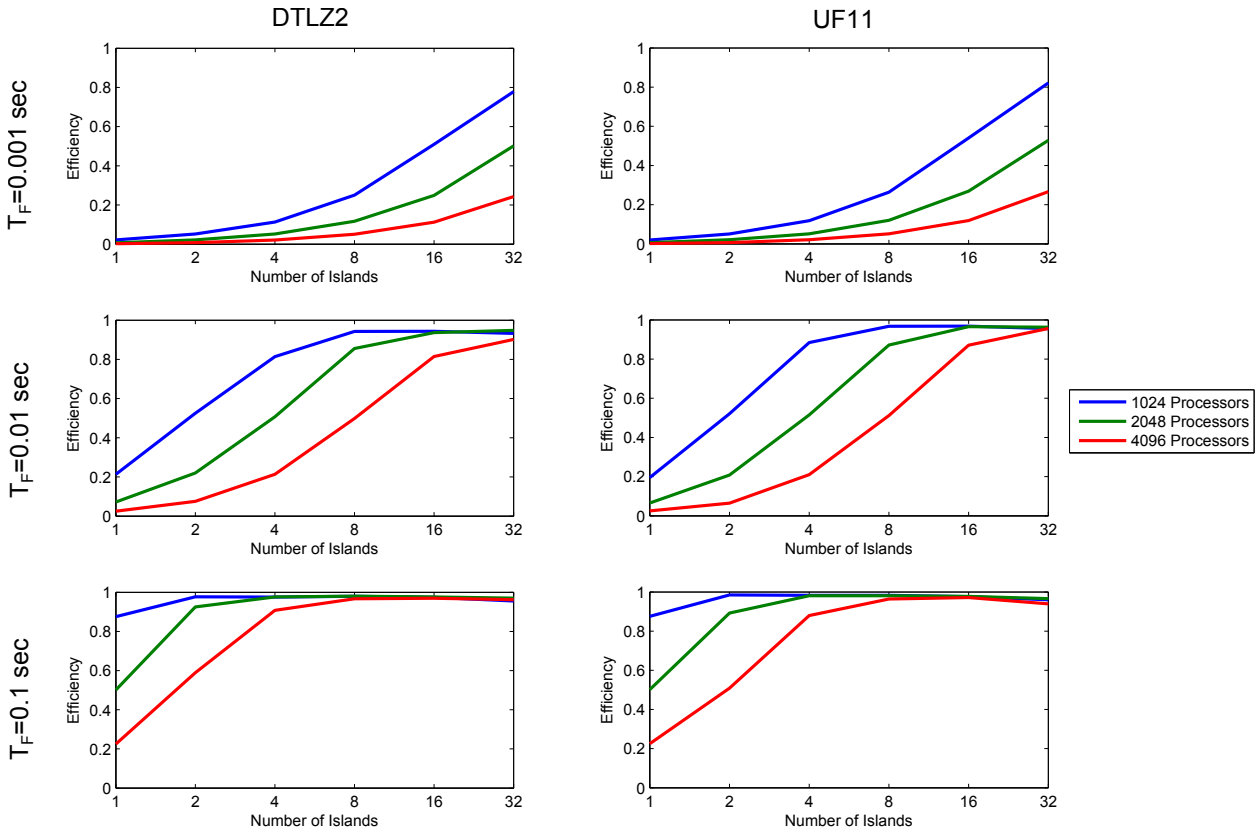


Figure 7.3: The average parallel efficiency of the multi-master Borg MOEA on the 5-objective DTLZ2 and UF11 test problems.

a consistent trend where increasing the number of islands increases the parallel efficiency of the multi-master Borg MOEA. When T_F is small, the parallel efficiency benefits from having more islands, since this allows the multi-master Borg MOEA to run each island with smaller, more efficient master-slave instances while reducing communication overhead. For $T_F = 0.01$ seconds, we observe that the parallel efficiency is maximized with 8 islands with 1024 processors, 16 islands with 2048 processors, and 32 islands with 4096 islands. For $T_F = 0.1$ seconds, the parallel efficiency is maximized with 2 islands with 1024 processors, 4 islands with 2048 processors, and 8 islands with 4096 processors. Again, we hypothesize that the hypervolume speedup of the multi-master Borg MOEA is maximized when the parallel efficiency is maximized.

Figures 7.4 and 7.5 show the hypervolume speedup for 5-objective DTLZ2 and UF11 test problems, respectively. These plots are similar to Figure 7.2, except the colored line series correspond to different numbers of islands. The tested processor counts are identified by each vertical column of subplots. Additionally, we are using the master-slave implementation as the baseline. Thus, a speedup of 4 indicates that multi-master configuration matched the hypervolume of the master-slave implementation in 1/4 the wallclock time.

While the $T_F = 0.001$ case in Figure 7.3 never reaches ideal parallel efficiency, our hypothesis would indicate that 32 islands will maximize hypervolume speedup since 32 islands offers the best parallel efficiency. This is confirmed in Figure 7.4 and Figure 7.5, which show 32 islands has the maximum hypervolume speedup for the tested processor counts. This demonstrates the advantages from using the multi-master Borg MOEA on problems with small T_F. For $T_F = 0.01$, hypervolume speedup is maximized with 8 islands with 1024 processors, 16 islands with 2048 processors, and 32 islands with 4096 processors. This matches exactly when the corresponding plots in Figure 7.3 indicate maximum parallel efficiency. Likewise, for $T_F = 0.1$, we see that hypervolume speedup is maximized with 2 islands with 1024 processors, 4 islands with 2048 processors, and 8 islands with 4096 processors. In Figure 7.5, the 4 island configuration provides slightly improved hypervolume speedup over the 8 island multi-master Borg MOEA. This closely matches when parallel efficiency is maximized in Figure 7.3. The difference observed between the hypervolume speedup on DTLZ2 and UF11 demonstrates how problem difficulty can influence the ideal parallel configuration. In this case, using fewer islands allows more NFE per island, which outperforms running more islands with fewer NFE per island.

These results confirm our hypothesis that maximizing the parallel efficiency of the Borg MOEA will maximize the hypervolume speedup. Consequently, maximizing parallel efficiency maximizes solution quality and convergence speed. This result supports prior results showing that the overall search quality of the Borg MOEA is only dependent on NFE and no other parameters (Hadka and Reed, 2012b).

7.2 Modeling the Parallel Borg MOEA

Parallel EAs can be classified into two categories: (1) synchronous and (2) asynchronous (Cantú-Paz, 2000). Synchronous EAs require that all of their population members are eval-

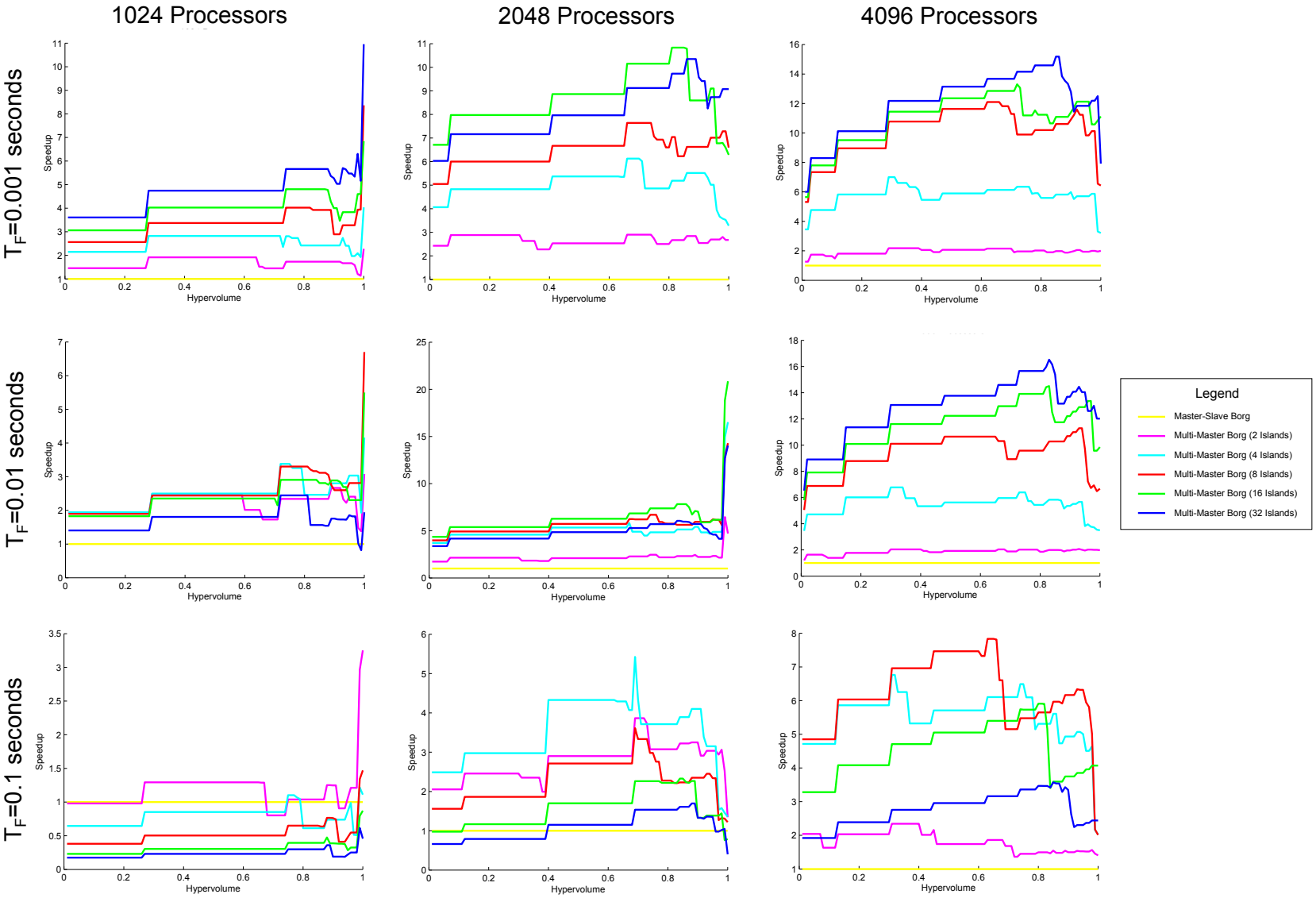


Figure 7.4: The average hypervolume speedup of the multi-master Borg MOEA on the 5-objective DTLZ2 test problem. The master-slave implementation is used as the baseline for computing hypervolume speedup.

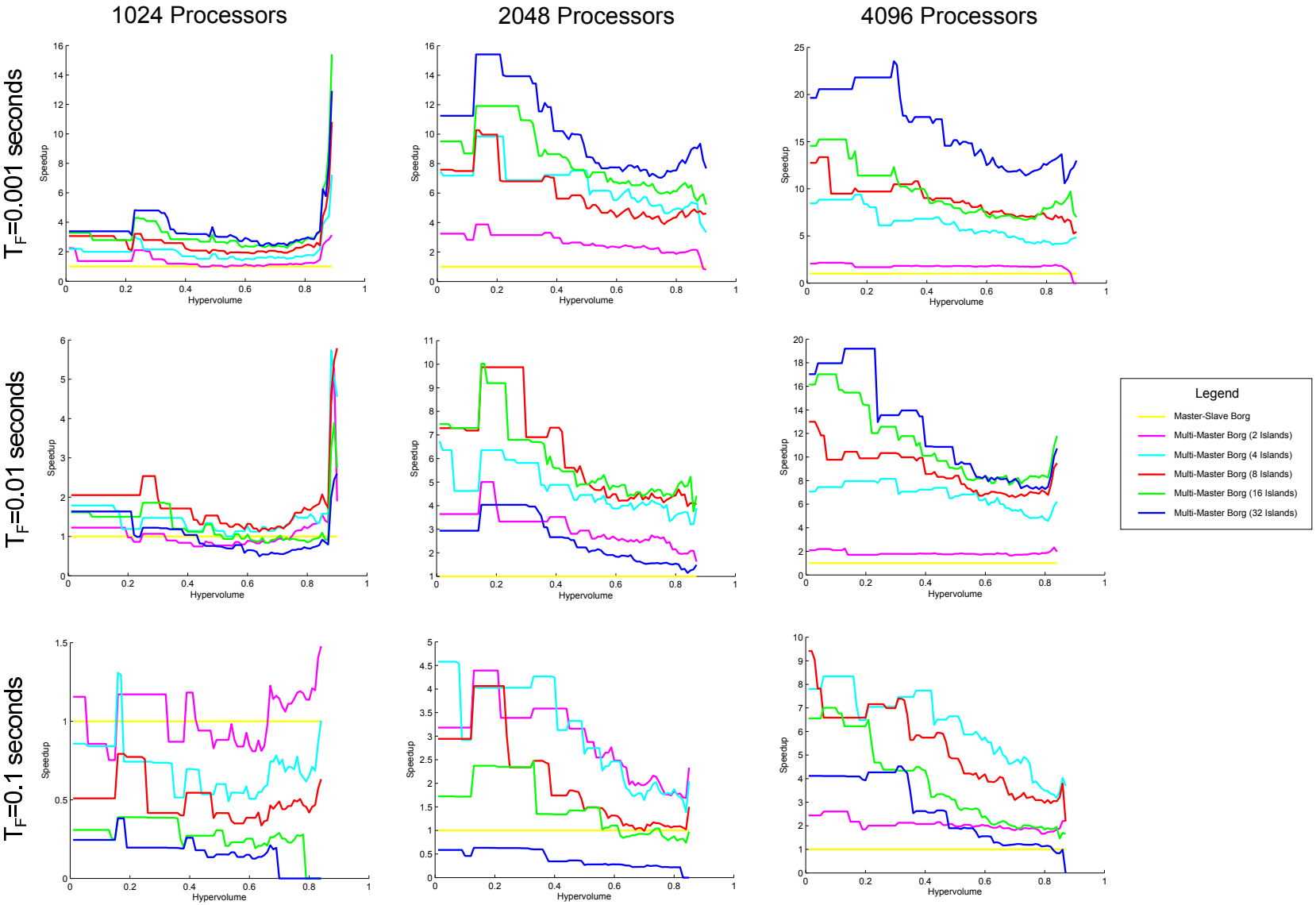


Figure 7.5: The average hypervolume speedup of the multi-master Borg MOEA on the 5-objective UF11 test problem. The master-slave implementation is used as the baseline for computing hypervolume speedup.

uated in a given generation before their evolutionary search proceeds to the next generation. Synchronization poses a computational bottleneck that leads to large algorithmic overhead and strongly limits the maximum parallelization efficiencies that can be achieved. Alternatively, asynchronous EAs avoid the synchronization step and eliminate their dependence on fully evaluated populations (i.e., generations), greatly expanding their potential to yield greater parallel efficiencies.

Since Bethke (1976) first attempted to parallelize a genetic algorithm in 1976, many studies have experimentally explored various methods of parallelizing EAs. For years, researchers developed different parallelization strategies and applied these algorithms successfully in many problem domains (Cantú-Paz, 1998). Then, in 1997, Erick Cantú-Paz began developing theoretical models for designing efficient parallel EAs (Cantú-Paz, 1997; Cantú-Paz and Goldberg, 1997a,b), which lead to his seminal publication (Cantú-Paz, 2000) detailing the theoretical properties of parallel EAs. Cantú-Paz's derivations focused on synchronous EAs, commenting only that asynchronous EAs would likely yield significant improvements in efficiency. However, to date, no detailed theoretical analysis of the scalability of asynchronous EAs has been published.

This section develops models for predicting the parallel behavior of the master-slave and multi-master Borg MOEA. Note that the parallel Borg MOEA is asynchronous, and these results can be generalized to typical asynchronous MOEAs. We explore the parallel scaling limits of the parallel Borg MOEA by developing an analytical model and a discrete event simulation model. The analytical model assumes a well-characterized, constant function evaluation time (T_F). This allows us to derive closed-form solutions for the expected speedup, efficiency, and processor count lower and upper bounds. The discrete event simulation simulation model allows us to better model communication costs, critical section overheads, and function evaluation times that follow probability distributions. Using these models, we show how we are able to more accurately model the parallel execution of the Borg MOEA.

The execution of the serial Borg MOEA consists of the following steps: (1) select and apply a search operator to produce offspring, (2) evaluate the offspring, (3) add the offspring to the population and archive, and (4) periodically check for stagnation (triggering a restart if necessary) and update the operator probabilities. For analysis, we model the time required to perform each of these execution steps using the notations T_C, T_F, and T_A. T_C is the time required to send and receive messages between the master and slave nodes. T_F is the time to evaluate one offspring. T_A is the time required to perform the serial components of the Borg MOEA, including adding the offspring to the population and archive, checking for stagnation, performing restarts, and adapting the operator probabilities. The total number of function evaluations allocated to the MOEA in a single run is denoted by N, and the number of processors available to the parallel algorithm is denoted by P. When P processors are available, one acts as a master node and $P - 1$ serve as slave nodes. This notation is summarized in Table 7.1.

Before beginning the scalability analysis, it is important to distinguish the difference between synchronous and asynchronous MOEAs. Most MOEAs in use today are generational, meaning that the population is evolved in distinct stages called generations. In a

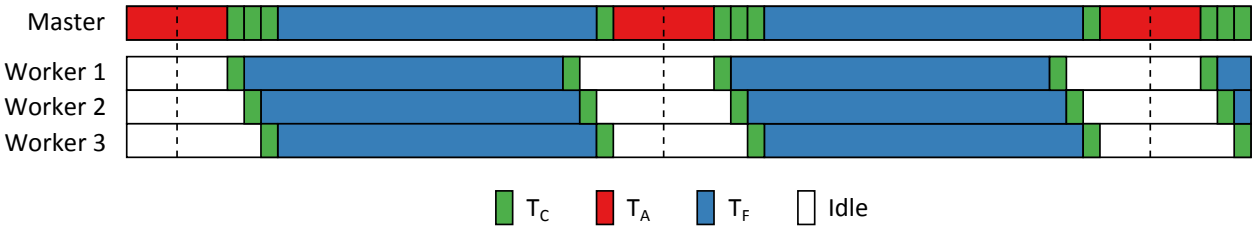


Figure 7.6: Diagram depicting the various costs incurred during a run of a synchronous, master-slave MOEA. In this example, $P = 4$ with one master and 3 slaves. The dotted line indicates the start of a new generation.

single generation, the population is evolved to produce offspring, the offspring are evaluated, and the offspring are added back into the population (possibly replacing existing members in the population). The key point here is that the previous generation must complete before the next generation starts. The need to synchronize generations gives rise to the term synchronous MOEA.

Figure 7.6 shows the timeline of events for a typical synchronous, master-slave MOEA. The vertical dotted lines indicate the start of a new generation. A generation begins when the master generates the offspring and sends them to the slave nodes for evaluation. Note that the master is also responsible for evaluating one offspring. It is also possible to send multiple solutions to a single slave node. In this study, however, we consider only the case where a single solution is sent. Next, the slaves evaluate the offspring and send the results back to the master node. Finally, the master updates the population with the offspring.

Asynchronous MOEAs eliminate the concept of a generation. As soon as an offspring is evaluated and returned to the master, the next offspring is immediately generated. This is shown in Figure 7.7. Note that as soon as the master receives the evaluated results from a slave, it immediately produces another offspring for that slave. Also note that the T_A for the asynchronous MOEA is shorter than the T_A for the synchronous MOEA. This is because the asynchronous MOEA processes one offspring at a time, whereas the synchronous MOEA processes all offspring from the generation at once. Comparing Figures 7.6 and 7.7, one can see the reduction in idle time using an asynchronous MOEA. In the remainder of this chapter, we develop and validate theoretical models for the master-slave and multi-master Borg MOEA. Since the parallel Borg MOEA is an asynchronous algorithm, the results presented here are generalizable to generic asynchronous MOEAs.

7.2.1 Runtime of the Serial Borg MOEA

In order to calculate speedup and efficiency, we need to derive the time for the serial algorithm. The total time for running a steady-state MOEA like the Borg MOEA in serial, denoted by T_S, is:

$$T_S = N\left(T_F + T_A\right) \tag{7.3}$$

The serial algorithm requires N function evaluations, where each function evaluation requires T_F and T_A, the time to generate the next offspring, the time to evaluate the offspring, and

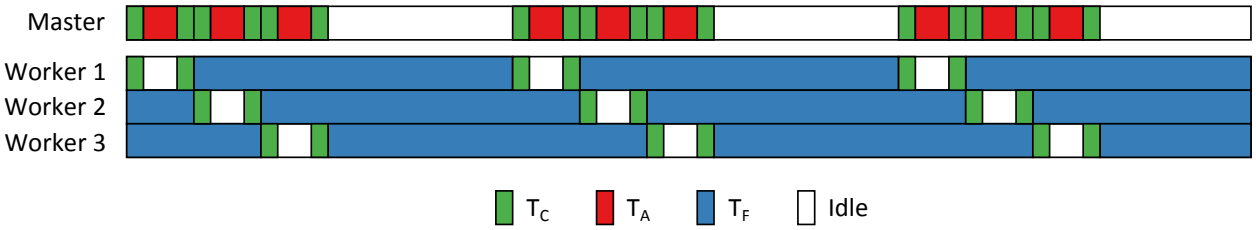


Figure 7.7: Diagram depicting the various costs incurred during a run of an asynchronous, master-slave MOEA. In this example, $P = 4$ with one master and 3 slaves. The master sends a solution to an available slave (T_C), the slave evaluates the solution (T_F), the slave sends the evaluated solution back to the master (T_C), and the master processes the solution and generates the next offspring to evaluate (T_A).

the time to process the evaluated offspring.

7.2.2 Runtime of the Master-Slave Borg MOEA

The parallel, master-slave implementation of the Borg MOEA follows a similar structure as the serial Borg MOEA. The only difference is the distribution of the evaluation of offspring to many slave nodes. Instead of generating and evaluating one offspring at a time, the master-slave implementation generates a new offspring whenever a slave node is available. Because the master-slave Borg MOEA is asynchronous and does not include any synchronization barriers, the slave nodes must compete with one another for access to the limited resources of the master node. To determine how well the master-slave Borg MOEA scales with increasing numbers of slaves, we start by building a simple, analytical model and work our way to a more accurate but complex simulation model.

Analytical Model

We begin our analysis by assuming that T_F, T_A, and T_C are constant. Assuming these times are constant allows us to model the asynchronous algorithm using an analytical model. Furthermore, by assuming all communication times are constant, all steps are performed in lockstep. The master node is guaranteed to be available when a slave node completes evaluating the objectives, eliminating any resource contention. This is seen in Figure 7.7, where the master node is always free to receive the result from a slave node as soon as the slave finishes evaluating the solution. From Figure 7.7, the parallel runtime of an asynchronous, master-slave MOEA is given by:

$$T_P = \frac{N}{P-1}\left(T_F + 2T_C + T_A\right) \tag{7.4}$$

One function evaluation requires $T_F + 2T_C + T_A$. This accounts for sending the decision variables to the slave node (T_C), evaluating the objectives (T_F), sending the objectives back to the master (T_C), processing the evaluated solution and generating the next offspring

(T_A). As the evaluations are spread across all slave nodes, each slave node performs $\frac{N}{P-1}$ evaluations.

One factor limiting scalability is the availability of the master node. We calculate the maximum number of processors that are feasible before the master reaches saturation (no idle time to process additional solutions) as:

$$P^{\text{UB}} = \frac{T_F}{2T_C + T_A} \tag{7.5}$$

Looking at this from a different perspective, we can ask how many processors are required to ensure the parallel implementation is at least faster than the serial implementation. To answer this question, we solve $\frac{T_S}{T_P} > 1$ to get:

$$P^{\text{LB}} > 2 + \frac{2T_C}{T_F + T_A} \tag{7.6}$$

From (7.6), we can calculate the lower bound on the number of processors needed for different time costs. Observe that the asynchronous model needs at least three processors to run faster than the serial algorithm regardless of the values of T_F, T_C, and T_A.

Discrete Event Simulation Model

The analytical model is limited by the assumptions that T_F, T_C, and T_A are constant. Relaxing these assumptions, we assume that T_F, T_C, and T_A follow a probability distribution. This introduces resource contention where the slave nodes must compete for access to the master node. When the master node is busy processing a request, slave nodes must wait in a queue until the master becomes available. This waiting will reduce the efficiency of the algorithm as P increases and resource contention becomes more likely. To model this more complex interaction, we build a discrete event simulation model.

The discrete event simulation model was developed in SimPy 2.3[1], a discrete event simulation library for Python. The structure of the simulation model is identical to that of the Borg MOEA. However, instead of actually performing the calculations or sending messages, the simulation model "holds" the resources for a set amount of time. For example, the master node would be modeled as follows. First, a "request" for the master simulates the slave waiting while the master is busy. Second, once the master is available, we "hold" the master to simulate the communication and algorithm processing time. Once the hold completes, the master is "released" and a slave is "activated". This release and activation is used to simulate sending a message to the slave node. This sequence of steps can be simulated in SimPy as follows.

```
1 yield request, self, master
2 yield hold, self, sampleTc() + sampleTa() + sampleTc()
3 yield release, self, master
```

[1] http://simpy.sourceforge.net/

```
activate(slave, slave.evaluate())
```

Accurate measurements of T_C, T_A, and T_F are needed for the simulation model to be accurate. Given a large sampling of these timing values for real executions of an algorithm on a parallel system, we used the R Project[2], an open-source language for statistical computing, to fit the sampled data to various distributions. Subsequently, the log-likelihood is calculated for each distribution to determine which best fits the sampled data. This is performed using the `fitdistr` function in the `MASS` library within the R Project, and selecting the distribution that produces the smallest log-likelihood value. For example, given the array of sampled times, we can fit the data to three different distribution as follows.

```
print(noquote("Normal:"))
normal <- fitdistr(times, "normal")
normal
normal\$loglik

print(noquote("Log Normal:"))
lognormal <- fitdistr(times, "lognormal")
lognormal
lognormal\$loglik

print(noquote("Exponential:"))
exponential <- fitdistr(times, "exponential")
exponential
exponential\$loglik
```

Running the resulting simulation model, we can compute the simulated runtime of the parallel algorithm, T_P. From this, we can predict the efficiency of the parallel algorithm with $E_P = \frac{T_S}{PT_P}$. The source code for this simulation is contained in Appendix C.

Validating the Simulation Model

On the Texas Advanced Computing Center (TACC) Ranger supercomputer, we were able to collect timing data with a resolution of 1 microsecond. This timing data was subsequently used to approximate the probability distributions of T_A and T_F. The value for T_C was captured separately by measuring the round-trip time to send and receive messages between the master and all slave nodes. This allows an accurate estimation of the point-to-point communication cost since the payload of each message is a constant size. On TACC Ranger, we calculated the value of T_C to be 6 microseconds.

Table 7.2 shows the predictions from the analytical and simulation model compared with experimental results collected from runs on TACC Ranger. The table shows the mean values

[2] `http://www.r-project.org`

for T_A, T_C, and T_F collected from the experiment. Using these time values, we can compute the predicted elapsed time from the analytical and simulation models. These predicted times are shown in Table 7.2 along with the relative error calculated with:

$$\text{Error} = \frac{\left|T_P^{\text{Actual}} - T_P^{\text{Predicted}}\right|}{\left|T_P^{\text{Actual}}\right|} \tag{7.7}$$

As hypothesized, the analytical model becomes error prone when the $\frac{T_F}{2T_C+T_A}$ ratio is small. This is seen by comparing the analytical model error as T_F increases. The error also increases as the processor count increases. This shows the fundamental limitation of the analytical model. It is unable to account for the resource contention encountered when a large number of slave nodes are attempting to communicate with the master node. This bottleneck is more accurately modeled by the simulation model, as indicated by the significantly lower error rates.

Also note in Table 7.2 the efficiency values recorded during this experiment. There is a clear peak in efficiency, where using fewer processors is underutilizing the system but using more processors increases resource contention. Consider how this compares to the processor count upper bound, which calculates the number of processors to saturate the master node. For demonstration, lets select the DTLZ2 case where $T_A = 0.000029$, $T_C = 0.000006$, and $T_A = 0.01$. From (7.5), the processor count upper bound is 244. However, as seen in Table 7.2, the peak efficiency occurs with approximately 32 processors. Maximizing the efficiency of the MOEA will require using fewer processors than the analytical model suggests.

This suggests that in situations where a large processor count is available and T_F is too small to run efficiently, better resource utilization may be possible with hierarchical topologies (Cantú-Paz, 2000; Coello Coello et al., 2007). Instead of running a single, large master-slave MOEA, the hierarchical topology runs several smaller, concurrently-running master-slave instances. Each of these instances runs on a distinct subset of the available processors. Our parallel performance simulation model can be used to determine the size of these subsets to maximize efficiency.

Comparison with Synchronous Model

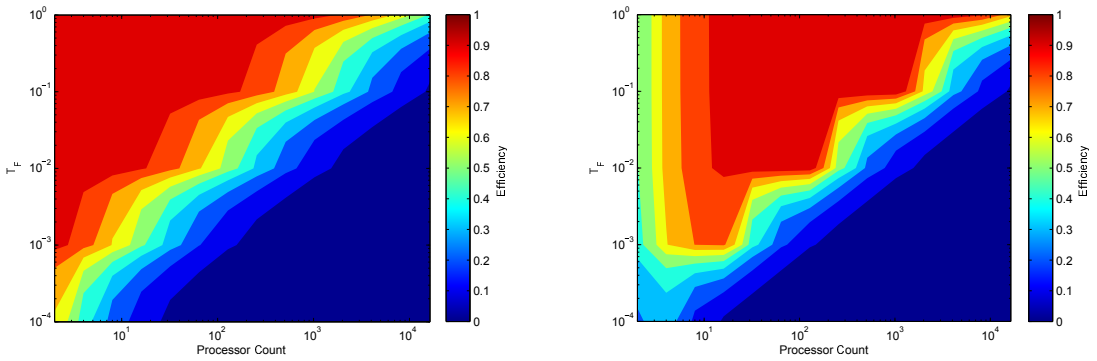
Building on our validated results for the simulation model, we now compare the scalability of the master-slave Borg MOEA to the synchronous MOEA developed by Cantú-Paz (Cantú-Paz, 2000). The analytical model developed by Cantú-Paz provides the following formula for the runtime of the parallel, synchronous MOEA:

$$T_P^{\text{Sync}} = \frac{N}{P}\left(T_F + PT_C + T_A^{\text{Sync}}\right) \tag{7.8}$$

Note that again we assume that each node processes only one solution per generation. Thus, P is both the processor count and population size. It is possible to have nodes evaluate more than one solution, potentially increasing efficiency when T_F and/or P is small. Cantú-Paz explores this in detail (Cantú-Paz, 2000). Also note that, in general, $T_A^{\text{Sync}} \approx PT_A$ since the synchronous algorithm has to process all P offspring at once.

Table 7.2: Table comparing the experimental results to the analytical and simulation models. All times are in seconds. Errors are percent deviation from experimental times.

					Experimental Results		Analytical Model		Simulation Model	
Problem	P	T_A	T_C	T_F	Time	Efficiency	Time	Error	Time	Error
DTLZ2	16	0.000023	0.000006	0.001	9.2	0.69	7.1	23%	7.2	22%
	32	0.000025	0.000006	0.001	6.3	0.51	3.5	45%	5.6	12%
	64	0.000027	0.000006	0.001	5.8	0.28	1.7	71%	6.0	4%
	128	0.000029	0.000006	0.001	6.3	0.13	0.9	86%	6.4	2%
	256	0.000031	0.000006	0.001	6.9	0.06	0.5	93%	6.8	2%
	512	0.000037	0.000006	0.001	7.9	0.03	0.3	97%	8.0	2%
	1024	0.000045	0.000006	0.001	9.4	0.01	0.2	98%	9.6	3%
	16	0.000023	0.000006	0.01	67.5	0.93	67.1	1%	67.1	1%
	32	0.000025	0.000006	0.01	33.1	0.95	32.5	2%	32.5	2%
	64	0.000027	0.000006	0.01	16.6	0.94	16.0	4%	16.0	4%
	128	0.000029	0.000006	0.01	8.8	0.89	8.0	10%	8.0	10%
	256	0.000031	0.000006	0.01	6.9	0.57	4.0	43%	6.8	2%
	512	0.000037	0.000006	0.01	7.8	0.25	2.0	75%	8.0	3%
	1024	0.000045	0.000006	0.01	9.4	0.10	1.0	90%	9.6	3%
	16	0.000023	0.000006	0.1	667.8	0.94	667.1	1%	667.4	1%
	32	0.000025	0.000006	0.1	323.1	0.97	322.8	1%	323.0	1%
	64	0.000027	0.000006	0.1	159.0	0.98	158.9	1%	159.0	0%
	128	0.000029	0.000006	0.1	79.0	0.99	78.8	1%	78.9	1%
	256	0.000031	0.000006	0.1	39.5	0.99	39.3	1%	39.3	1%
	512	0.000037	0.000006	0.1	19.9	0.98	19.6	2%	19.7	2%
	1024	0.000045	0.000006	0.1	11.5	0.85	9.8	15%	10.0	14%
UF11	16	0.000055	0.000006	0.001	12.3	0.54	7.5	40%	11.6	6%
	32	0.000057	0.000006	0.001	11.2	0.29	3.7	67%	12.0	8%
	64	0.000059	0.000006	0.001	11.5	0.14	1.8	85%	12.4	8%
	128	0.000061	0.000006	0.001	11.8	0.07	0.9	93%	12.4	6%
	256	0.000064	0.000006	0.001	13.3	0.03	0.5	97%	13.4	1%
	512	0.000068	0.000006	0.001	14.2	0.01	0.3	98%	14.2	0%
	1024	0.000078	0.000006	0.001	16.3	0.01	0.2	99%	16.1	2%
	16	0.000055	0.000006	0.01	68.5	0.92	67.5	2%	67.6	2%
	32	0.000057	0.000006	0.01	35.2	0.89	32.7	8%	32.8	7%
	64	0.000059	0.000006	0.01	18.4	0.85	16.1	13%	16.3	12%
	128	0.000061	0.000006	0.01	12.6	0.62	8.0	37%	12.4	2%
	256	0.000064	0.000006	0.01	13.4	0.29	4.0	71%	13.4	0%
	512	0.000068	0.000006	0.01	14.2	0.14	2.0	86%	14.2	0%
	1024	0.000078	0.000006	0.01	16.2	0.06	1.0	94%	16.1	1%
	16	0.000055	0.000006	0.1	668.7	0.94	667.5	1%	667.8	1%
	32	0.000057	0.000006	0.1	323.4	0.97	323.0	1%	323.2	1%
	64	0.000059	0.000006	0.1	159.3	0.98	159.0	1%	159.0	1%
	128	0.000061	0.000006	0.1	79.2	0.99	78.9	1%	79.0	1%
	256	0.000064	0.000006	0.1	39.8	0.98	39.3	2%	39.4	2%
	512	0.000068	0.000006	0.1	20.8	0.94	19.6	6%	19.7	6%
	1024	0.000078	0.000006	0.1	16.6	0.59	9.8	41%	16.3	2%



(a) Synchronous Efficiency

(b) Asynchronous Efficiency

Figure 7.8: Predicted efficiency of a synchronous MOEA (using the model developed by Erick Cantú-Paz (Cantú-Paz, 2000)) compared against the predicted efficiency of an asynchronous MOEA using the simulation model. T_F ranges from 0.0001 up to 1 second, and P ranges from 2 to 16,384 processors. The coloring shows the efficiency, with highest efficiency in the red regions and worst efficiency in the blue regions. Note the log scale of the x- and y- axes.

Figure 7.8 shows the predicted efficiency from both models across a range of T_F and P values. T_F ranges from 0.0001 up to 1 second, and P ranges from 2 to 16384 processors. Complex engineered systems design has been strongly limited by computational barriers where evaluation times greatly exceed 1 second or more (Bloebaum and McGowan, 2010), so understanding scaling limits with large T_F and P is important. For both models, T_A and T_C are fixed at 0.000060 and 0.000006 seconds, respectively.

Note that the synchronous MOEA is able to achieve higher efficiency with smaller T_F and P. The asynchronous model appears to have a lower bound processor count of 16 and a lower bound T_F of 0.01 seconds. However, the asynchronous model is able to scale to larger processor counts than the synchronous model with the same T_F. As discussed, this is a result of the asynchronous model not requiring synchronization barriers at each generation. This provides the first theoretical results that explain in detail the conditions necessary for the asynchronous model to efficiently scale to larger processor counts than the commonly used synchronous model.

Another substantial difference between the synchronous and asynchronous MOEA is the impact on performance of highly-variable function evaluation times (T_F). Since the synchronous MOEA must wait for all slaves to complete each generation, all nodes sit idle waiting for the longest running evaluation to complete. The asynchronous MOEA, on the other hand, is able to immediately send another offspring to a slave as soon as it finishes the previous evaluation. So, when T_F is highly variable, we expect the efficiency of the synchronous model to decline while the asynchronous model remains unchanged.

7.2.3 Runtime of the Multi-Master Borg MOEA

Recall from Chapter 6 that the multi-master Borg MOEA is a hierarchical extension of the master-slave Borg MOEA. Each of the islands in the multi-master Borg MOEA runs an independent instance of the master-slave Borg MOEA. Therefore, the model developed in Section 7.2.2 can be extended to predict the efficiency of the multi-master Borg MOEA. The only additional overhead of the multi-master Borg MOEA is the messages between the master nodes and the controller.

The master has two communication patterns with the controller. First, the master sends periodic updates of the recently-discovered ϵ-dominant solutions to the controller. With its default settings, these update messages are sent every 10000 NFE. Second, when the master is struggling, it requests help and receives guidance from the controller. These messages are more difficult to model as they occur unpredictably.

Additionally, these two messages introduce different loads onto the master and controller node. During an update, the controller must add the transferred solutions to the global ϵ-dominance archive. During a help / guidance message, the master must add the solutions contained within the guidance packet to its local ϵ-dominance archive. Inserting k solutions into an ϵ-dominance archive has a runtime complexity of $O(kMN)$, where M is the number of objectives and N is the size of the ϵ-dominance archive. This tends to be significantly more expensive than the transmission of those solutions between the nodes. As an example, on a high performance network such as the one used by the TACC Ranger system, transmitting 250 solutions takes approximately 0.00025 seconds. Adding 250 solutions to an ϵ-dominance archive introduces an overhead of approximately 0.01 seconds.

This implies that updates are cheap for masters but expensive for the controller, and help / guidance messages are expensive for masters but cheap for the controller. Since help / guidance messages are only used when a master is struggling, we claim the overhead incurred by the master is acceptable since the outcome of a successful global restart benefits the algorithm. This leaves the update messages as the remaining potential bottleneck.

We can apply queueing theory to model the overhead experienced by the controller node as the number of islands increases (Heidelberger and Trivedi, 1982). This is accomplished using an M/M/1 queue. The M/M/1 queue requires estimating the interarrival rate, which is the frequency that new jobs arrive in the queue, and the service rate, which is the frequency that queued jobs are processed and removed from the queue. The two "M"s in the name M/M/1 queue indicate that the interarrival and service rates are modeled as Markov processes. The interarrival rate is based on the update frequency, which by default is 10000 NFE, the evaluation time of solutions ($T_F + 2T_C + T_A$), and the number of master nodes. The service rate is based on the number of solutions in the update message and how quickly they can be processed by the controller node. Given these two rates, we can calculate the average wait time with

$$\text{average wait time} = \frac{\lambda/\mu}{\mu - \lambda}, \tag{7.9}$$

where λ is the interarrival rate and μ is the service rate. If $\mu > \lambda$, then the average wait time is undefined since the queue will grow indefinitely.

Table 7.3: The average wait time of messages in the controller.

Islands	$T_F = 0.001$	$T_F = 0.01$	$T_F = 0.1$
2	1.86×10^{-5}	1.98×10^{-6}	1.99×10^{-7}
4	3.74×10^{-5}	3.97×10^{-6}	3.99×10^{-7}
8	7.51×10^{-5}	7.94×10^{-6}	7.99×10^{-7}
16	0.000151	1.59×10^{-5}	1.59×10^{-6}
32	0.000307	3.18×10^{-5}	3.19×10^{-6}
64	0.000634	6.39×10^{-5}	6.39×10^{-6}
128	0.001355	0.000128	1.28×10^{-5}
256	0.003137	0.000260	2.56×10^{-5}
512	0.009142	0.000535	5.14×10^{-5}
1024	0.213333	0.001131	0.000103

As an example, suppose we estimate the service rate to be $\mu = 0.01$, $T_C = 0.000006$, and $T_A = 0.00006$ seconds (these times are representative of numerical test problems like DTLZ2 and UF11). The calculated average wait times for various T_F are shown in Table 7.3. For the small island counts, the controller is unburdened. Only when T_F is small and the number of islands is large does the average wait time begin to become significant. On many real-world problems where $T_F \geq 0.1$, the multi-master Borg MOEA can easily support one thousand islands. Additionally, the controller is not a bottleneck except when T_F is extremely small and the number of islands is extremely large. We can therefore use the master-slave discrete event simulation model to predict the efficiency of individual islands when the average wait time of the controller is small.

7.3 Ideal Configuration

In this chapter, we explored the impact of the configuration of the parallel Borg MOEA on search quality. In Section 7.1, we identified a strong correlation between parallel efficiency and hypervolume speedup. To maximize hypervolume speedup (i.e., solution quality), we want to maximize the parallel efficiency of the Borg MOEA. Then, in Section 7.2 we built an analytical and discrete event simulation model for predicting the efficiency of the parallel Borg MOEA. With these tools, we propose the following strategy for configuring the Borg MOEA.

First, we determine the ideal processor count for the master-slave Borg MOEA. This is accomplished using the proposed discrete event simulation model. This requires the collection of timing data to estimate T_A and T_C. Given these time estimates, running the discrete event simulation model with various processor counts will determine which processor count, P^*, yields the maximum predicted parallel efficiency. If P^* is greater than the number of available processors, then it is recommended to run the master-slave Borg MOEA on all available processors. While the master-slave Borg MOEA will not run at maximum efficiency, it will

run at the maximum achievable efficiency given the available processors. If P^* is less than the number of available processors, then is is recommended to run the multi-master Borg MOEA on the largest number of processors that is a multiple of P^*, with the number of islands equal to this multiple. This ensures that each island in the multi-master is running with P^* processors with the maximum predicted parallel efficiency.

7.4 Conclusion

In this study, we analyzed the scalability of the master-slave and multi-master Borg MOEA. We first experimentally analyzed the parallel efficiency (naive speedup) and hypervolume speedup of various configurations of the parallel Borg MOEA. In doing so, we identified the correlation between maximizing parallel efficiency and maximizing hypervolume speedup, a result that corroborates the analysis from Chapter 4 and Chapter 5 that demonstrated Borg's key parametric sensitivity to NFE.

Next, we developed an analytical model of the parallel processing time and derived the processor count lower and upper bounds. This analytical model is limited by its inability to model the interactions between the master and slave nodes that introduce resource contention and additional overhead. To more accurately model the parallel Borg MOEA, we developed a discrete event simulation model using the SimPy simulation package for Python. From this model, we can accurately model the parallel processing time, efficiency, and ideal processor count to maximize efficiency.

Finally, we propose a strategy for using the discrete event simulation model to configure the parallel Borg MOEA for specific problems. This strategy aims to maximize the hypervolume speedup by maximizing the efficiency of the parallel Borg MOEA.

Chapter 8

Case Study: Risk-Based Water Supply Portfolio Planning

This chapter is drawn from the following paper: "Hadka, D., Reed, P.M., and Madduri, K. *Large-Scale Parallelization of the Borg MOEA for Addressing the Design of Complex Engineered Systems.* Evolutionary Computation, In Review."

This chapter explores the application of the Borg MOEA on a real-world complex engineered system: a severely constrained, six objective risk-based water supply portfolio planning problem called the LRGV problem (Kasprzyk et al., 2009, 2011). This problem features many of the challenging problem properties discussed in Chapter 6. It is many-objective, multi-modal, non-linear, contains a mix of discrete and real decision variables, is severely constrained, and has stochastic objectives with expensive function evaluation times. In Reed et al. (2012), all of the tested state-of-the-art MOEAs struggle to solve this problem reliably, including the serial Borg MOEA. Using this problem, we demonstrate that the parallel variants of the Borg MOEA developed in Chapter 6 significantly improve speed of convergence, solution quality, and reliability.

The remainder of this chapter is organized up as follows. Section 8.1 introduces the LRGV problem. Section 8.2 discusses the experimental setup of this study where the two parallel variants of the Borg MOEA are run on the TACC Ranger system. Section 8.3 presents the results from this parallel analysis. Finally, Section 8.4 summarizes the findings of this chapter.

8.1 Introduction

This section introduces a challenging complex engineered system used to test the effectiveness, efficiency, and reliability of the parallel variants of the Borg MOEA. Urban water supply management is the act of securing and allocating water resources to a locale under varying environmental and economic conditions. Population growth, increased urbanization, water scarcity due to droughts, and climate change are factors that challenge water supply management and increase the risk of critical water supply failures (Kundzewicz et al., 2007;

Frederick and Schwarz, 1999; Lane et al., 1999; Vorosmarty et al., 2000; Milly et al., 2008; Brekke et al., 2009). A number of approaches can be taken to facilitate increases in demand and mitigate the impact of supply fluctuations. The municipality can undertake structural improvements, such as building new reservoirs, and non-structural adaptations, such as purchasing water on water markets (Anderson and Hill, 1997). Water markets aim to allocate water resources to their highest-value use by transferring volumes of water across regions or user sectors (Israel and Lund, 1995; Hadjigeorgalis, 2008).

In this case study, water supplies can be purchased using three market mechanisms: permanent rights, leases, and options. Permanent rights represent the purchase of a fixed percentage of the stream inflows to a reservoir. Leases facilitate short-term transfers of water from agricultural users to a city, but prices fluctuate with supply and demand. For instance, the onset of drought conditions can lead to a spike in prices. Alternatively, options reserve volumes of water at a fixed price that can be transferred later in the year. Options that remain unused at the end of the year are dropped, and can become costly if the city holds many unused options at the end of the year.

Several studies considering only single-objective formulations of this problem have shown that water markets with both options and leases can reduce the overall cost associated with maintaining reliable urban water supplies (Lund, 1995; Wilchfort and Lund, 1997; Watkins Jr. and McKinney, 1999; Jenkins and Lund, 2000; Characklis et al., 2006; Kirsch et al., 2009). Kasprzyk et al. (2009) proposed the first many-objective formulation of this problem, allowing tradeoffs between cost, reliability, surplus water, cost variability, frequency of using leases, and unused transfers of water. They applied this problem to a city located in the Lower Rio Grande Valley (LRGV) in southern Texas with a 10-year planning horizon. A Monte Carlo simulation models the city using both thirty-three years of historical data from the region with additional factors like growing population demands, variable hydrologic conditions, and market pricing dynamics. In this study, we use the most challenging "Case D" variant of the problem from Kasprzyk et al. (2009) and refer to it as the LRGV problem.

The LRGV problem consists of 8 decision variables, 6 objectives, and 3 constraints. The 8 decision variables shown in Table 8.1 control the use of permanent rights, options, and leases by the simulation model. Several of these decision variables are discrete. Since the Borg MOEA uses real-valued operators, the decision variables are rounded to the nearest integer prior to invoking the simulation model. The simulation model outputs the 6 objectives shown in Table 8.2. The LRGV problem is thus defined by

$$F(x) = (f_{\text{cost}}(x), f_{\text{rel}}(x), f_{\text{surplus}}(x), f_{\text{costvar}}(x), f_{\text{dropped}}(x), f_{\text{leases}}(x)) \tag{8.1}$$

where

$$x = (N_R, N_{O_{\text{low}}}, N_{O_{\text{high}}}, \xi, \alpha_{\text{May-Dec}}, \beta_{\text{May-Dec}}, \alpha_{\text{Jan-Apr}}, \beta_{\text{Jan-Apr}}). \tag{8.2}$$

The 3 constraints ensure that potential solutions satisfy limits in cost variability, reliability, and critical reliability. Reliability measures small failures that can be mitigated by water conservation or other practices. Critical reliability measures larger failures where the city fails to meet more than 60% of the required demand in a given month. Formally, these

Table 8.1: Decision variables used by the LRGV problem.

Decision Variable	Type	Range	Description
N_R	Integer	30,000-60,000	Volume of permanent rights
$N_{O_{\text{low}}}$	Integer	0-20,000	Low-volume options contracts
$N_{O_{\text{high}}}$	Real	$N_{O_{\text{low}}} - 2N_{O_{\text{low}}}$	High-volume options contracts
ξ	Real	0.1-0.4	Low to high options threshold
$\alpha_{\text{May-Dec}}$	Real	0.0-3.0	Lease/options strategy for May-Dec ("when to acquire")
$\beta_{\text{May-Dec}}$	Real	$\alpha_{\text{May-Dec}}$-3.0	Lease/options strategy for May-Dec ("how much to acquire")
$\alpha_{\text{Jan-Apr}}$	Real	0.0-3.0	Lease/options strategy for Jan-Apr ("when to acquire")
$\beta_{\text{Jan-Apr}}$	Real	$\alpha_{\text{Jan-Apr}}$-3.0	Lease/options strategy for Jan-Apr ("how much to acquire")

Table 8.2: Objectives used by the LRGV problem.

Objective	Description	Direction	ϵ Search Precision
f_{cost}	Cost	Min	0.003
f_{rel}	Reliability	Max	0.002
f_{surplus}	Surplus	Min	0.01
f_{costvar}	Cost Variability	Min	0.001
f_{dropped}	Dropped Transfers	Min	0.002
f_{leases}	Number of Leases	Min	0.003

constraints are defined by

$$f_{\text{costvar}} < 1.1 \tag{8.3}$$

$$f_{\text{rel}} > 0.98 \tag{8.4}$$

$$Pr[S_{i,j} > 0.6d_{i,j}] = 1.0,\ \forall i \in [1, 12] \text{ and } j \in [1, T] \tag{8.5}$$

where $S_{i,j}$ is the simulated supply and $d_{i,j}$ is the simulated demand for month i in the year j, and $T = 10$ is the number of simulated years. Full details of the LRGV problem are available in Kasprzyk et al. (2009, 2011).

Since the LRGV simulation is stochastic, many Monte Carlo trials are performed when computing the expected values for its performance objectives. Increasing the number of Monte Carlo trials will improve the quality the estimates of the expected values for the objectives, but also significantly increases the evaluation time. In this study, 1000 samples are used, resulting in an evaluation time of approximately 0.14 seconds.

The first attempts to solve the LRGV problem used the ϵ-NSGA-II to discover the trade-offs between various market strategies and their impact on cost and reliability when faced with the uncertainty and risks inherent in water portfolio planning (Kasprzyk et al., 2009). Reed et al. (2012) performed a rigorous assessment of several MOEAs on the LRGV problem, identifying that all of the top serial MOEAs struggled with their attainments and controllability, many of which completely failed on this problem.

These search failures are the result of several problem characteristics. First, the LRGV problem is a many-objective problem with a fully stochastic objective space. Many MOEAs are unable to cope with problems with four or more deterministic objectives as they are unable to effectively navigate and search high-dimensional spaces (Purshouse and Fleming, 2003, 2007; Hadka and Reed, 2012b). Second, the problem is severely constrained. Reed et al. (2012) showed a random sampling baseline where the probability of randomly generating a feasible solution for the LRGV problem is approximately 1 in 500000. This implies the initial population will likely consist entirely of infeasible solutions, requiring the MOEA to direct search towards feasible regions. MOEAs unable to do so will fail to generate any Pareto approximate solutions. Third, as identified in Kasprzyk et al. (2009), the best-known reference set consists of three disjoint regions corresponding to vastly different water planning strategies. A successful MOEA must be able to locate and diversify across all disjoint regions within the best known Pareto approximate set. Finally, the LRGV problem has an expensive function evaluation time. As mentioned previously, the objective function evaluation time in this study is approximately 0.14 seconds. This necessitates the use of parallel MOEAs in order to discover high-quality solutions in a reasonable amount of wallclock time.

8.2 Methodology

This study compared the master-slave and multi-master Borg MOEA implementations against the ϵ-NSGA-II algorithm originally used to explore the LRGV problem. ϵ-NSGA-II is one of the top-performing MOEAs on the LRGV problem (Reed et al., 2012). In this study,

Table 8.3: The parallel MOEAs tested in this study and their salient characteristics.

Implementation	Islands	Initialization	Style	Operator
Master-Slave ϵ-NSGA-II	1	Uniform	Generational	SBX+PM
Master-Slave Borg	1	Latin	Steady-State	Multi-operator
Multi-Master Borg	2	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	4	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	8	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	16	Global Latin	Steady-State	Multi-operator
Multi-Master Borg	32	Global Latin	Steady-State	Multi-operator

we are using the large-cluster master-slave ϵ-NSGA-II implementation from Reed et al. (2008). The master-slave and multi-master Borg MOEA implementations were written in high-performance C with the use of MPI to facilitate communication between nodes. This code was compiled and executed on the Texas Advanced Computing Center (TACC) Ranger system. TACC Ranger consists of 3,936 16-way symmetric multiprocessing (SMP) compute nodes, each containing four 2.3 GHz AMD Opteron Quad-Core 64-bit processors and 32 GBs of memory. Each core can perform 9.2 GFLOPS. In total, there are 62976 processing cores. Throughout this dissertation, we refer to these individual processing cores as "processors". Nodes are connected using two large Sun InfiniBand DataCenter switches.

The master-slave and multi-master Borg MOEA implementations were executed in a number of different configurations to compare their scalability and solution quality at large processor counts. On TACC Ranger, submissions are limited to 16384 cores. Therefore, the three implementations were each executed with 1024, 2048, 4096, 8192, and 16384 cores. Additionally, the multi-master runs used different topologies with 2, 4, 8, 16 and 32 islands. A single run of an implementation was given 10 minutes of wallclock time, and allowed to evaluate as many objective function evaluations as it could manage. Each run was repeated 50 times with different initial random seeds so that the expected search quality and its deviation can be calculated. A summary of the algorithms tested in this study are given in Table 8.3.

The output of each run is the approximation set generated by the algorithm. This approximation set is stored in a database. After all runs have been executed, the aggregation of all approximation sets across all algorithms forms the reference set. This reference set contains all Pareto approximate solutions discovered in this study. Using this reference set, we can subsequently compute various performance indicators. Based our prior comprehensive assessment of the LRGV test case for a broad suite of MOEAs (Reed et al., 2012), we have selected to emphasize the hypervolume indicator. Our prior results have shown that the hypervolume is sensitive to the irregular Pareto approximate set geometry of the LRGV test case and that, in general, other measures are equivalent or easier to satisfy at high levels of performance. Hypervolume measures the volume of objective space dominated by an approximation set. Larger hypervolumes therefore correspond to approximation sets that dominate more space, which in turn indicates high-quality approximation sets.

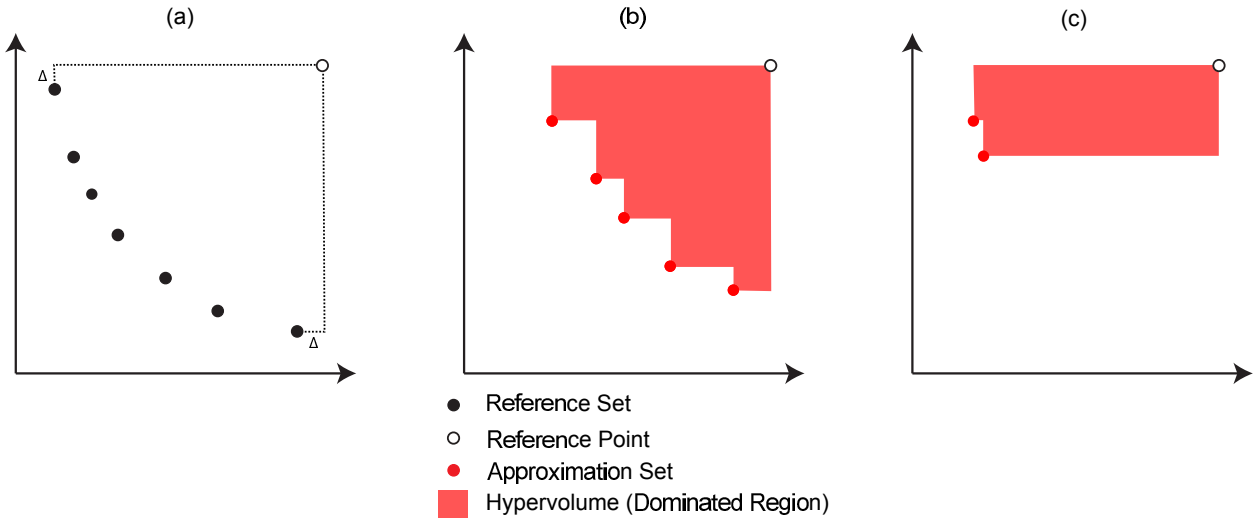


Figure 8.1: 2D demonstration of the hypervolume indicator. (a) The bounds of the reference set are used to calculate the reference point; this calculation typically adds a delta so that the boundary points contribute positive hypervolume. (b) Given an approximation set, the hypervolume is the volume of space dominated between the approximation set points and the reference point. (c) Demonstration of how an approximation set with good proximity but poor diversity results in a sub-optimal hypervolume.

Figure 8.1 shows an example of how hypervolume is computed in 2D space. A reference point is chosen based on the bounds of the reference set plus some additional delta. This delta ensures the boundary points contribute positive volume to the overall hypervolume. Hypervolume is normalized to the range $[0, 1]$ such that the best possible set, the reference set, has a hypervolume of 1. Approximation sets with hypervolumes near 1 are high-quality, have converged in proximity to the reference set, and are diversified across the entire Pareto front.

While hypervolume can be expensive to calculate, it offers several advantages over other performance indicators. Its results are scaling independent, it is compatible with the dominance relation, and its meaning is intuitive (Zitzler et al., 2002c). Since the LRGV problem has six objectives, we elected to use the efficient WFG hypervolume algorithm to calculate exact hypervolume values (While et al., 2012).

In addition to recording the end-of-run approximation set, runtime data is collected every 10,000 NFE and stored in the database. The data includes a snapshot of the approximation set discovered by the algorithm at the current point in time, the operator probabilities used by the Borg MOEA's adaptive multi-operator mechanism, and local and global restart frequencies. Identical to how we compute hypervolume for the end-of-run approximation set, we also compute hypervolume for each snapshot. This provides a view into the dynamics of the algorithm. We can visualize the inner workings of the parallel Borg MOEA and its

impact on solution quality.

8.3 Results

The LRGV problem described in Section 8.1 was solved using the large-cluster master-slave ϵ-NSGA-II, the master-slave Borg MOEA, and several configurations of the multi-master Borg MOEA as described in Section 8.2. This section presents the results from this experiment. First, Section 8.3.1 investigates the time required to converge to high-quality solutions, identifying the implementations which converged fastest and with the highest reliability. Second, we explore the end-of-run solution quality as a result of running each implementation for a fixed amount of time in Section 8.3.2, identifying the implementation that produced the highest-quality result. In Section 8.3.3, we analyze the operator dynamics introduced by the auto-adaptive multi-operator search mechanism used by the Borg MOEA. Finally, Section 8.3.4 calculates the parallel efficiency and speedup of the implementations, identifying the configurations that maximize their use of the underlying computing resources.

8.3.1 Convergence Speed and Reliability

Figure 8.2 shows the speed and reliability of the different parallel MOEA implementations tested in this study. These results show the cumulative distribution functions (CDFs) for generating high-quality approximation sets with respect to wallclock time. Here, an algorithm generates a high-quality approximation set if its hypervolume is $\geq 90\%$ of the best-known, reference set hypervolume. Each of the subplots in Figure 8.2 shows the results for different processor counts. Each of the line series corresponds to one of the implementations in Table 8.3. These line series plot at each point in time the probability that the implementation generated high-quality approximation sets exceeding the 90% hypervolume threshold. Ideal performance on these plots are vertical CDFs (i.e., no random seed variability) at a minimum wallclock.

Starting with the 1024 processor subplot, we observe that none of the implementations had a 100% probability of attaining the 90% hypervolume threshold within the wallclock allowed. The closest results were provided by the 16 and 32 island multi-master Borg MOEA implementations, which reached the hypervolume threshold with 90% probability. This is followed closely by the 8 island multi-master Borg MOEA implementation with 85% probability, and more distantly by the 2 and 4 island multi-master Borg MOEA implementations with 60% and 55% probability, respectively. The high failure rates for several configurations of the parallel Borg MOEA confirm the difficulty of the LRGV case study as has been observed in prior work (Reed et al., 2012). All of the multi-master Borg MOEA implementations significantly exceeded the reliability of the master-slave Borg MOEA and ϵ-NSGA-II implementations. Note that the slopes of all of the success rate CDFs show strong random seed variability in the time required to attain high-quality approximations of the LRGV case study's tradeoffs.

Additionally, by observing the position along the x-axis where the line series reached

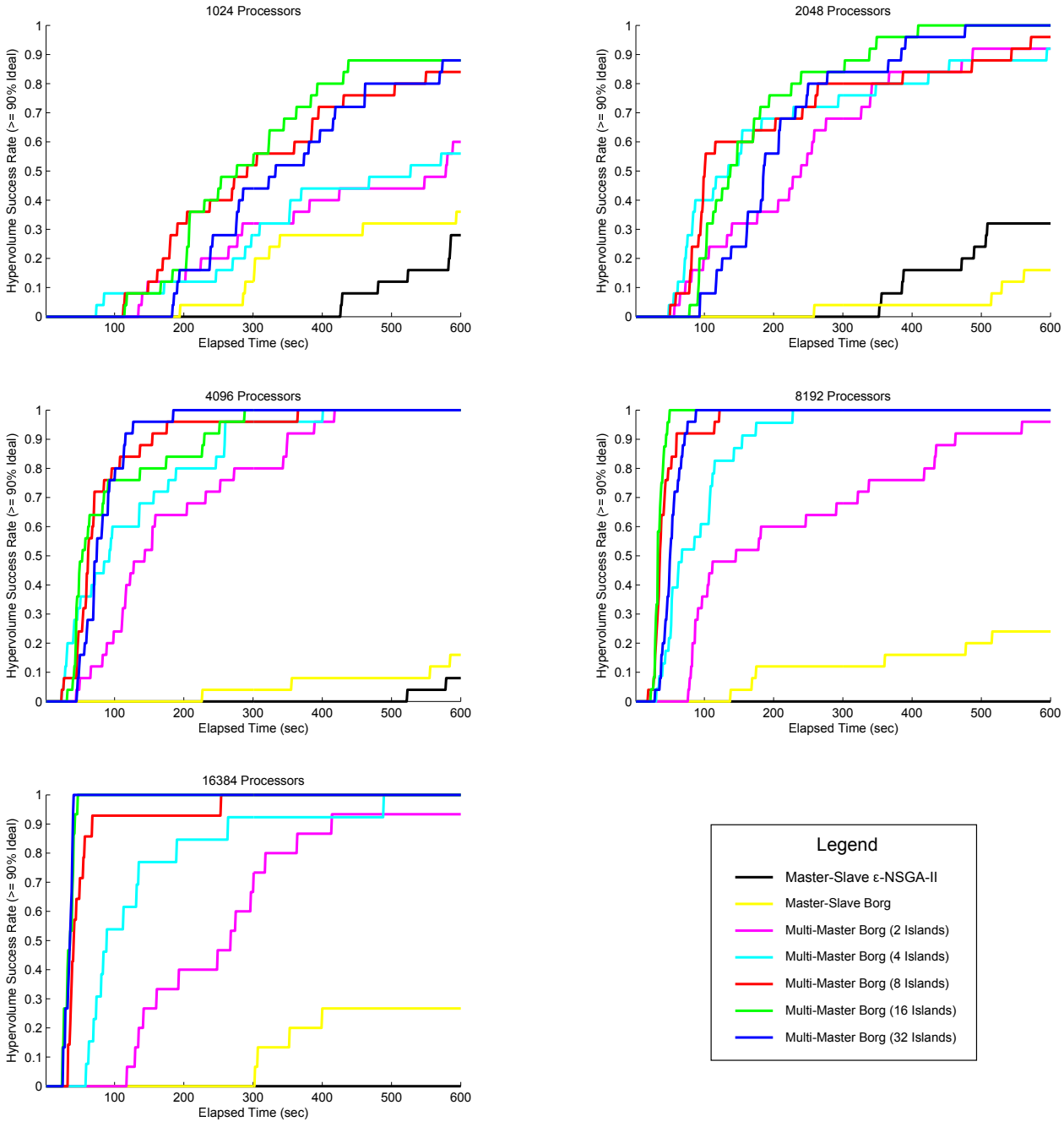


Figure 8.2: Probability of each parallel implementation of attaining a hypervolume >= 90% of the reference set hypervolume on the LRGV problem. Each subplot shows the results for different processor counts, from 1024 up to 16384 processors.

their maximum, we can determine the convergence speed of the algorithm. Continuing with our analysis, we observe in the 1024 processor subplot that the 16 and 32 island multi-master Borg MOEA implementations converged in 450 and 560 seconds, respectively. In general, we desire MOEAs that produce the highest-quality results. As with this case, when the quality attained by two different implementations are equivalent, we then look at the speed of convergence. For the 1024 processor case, the 16 island multi-master Borg MOEA implementation produced the best result.

As the processor count increases, we observe that many implementations are able to reach the 90% hypervolume threshold with 100% probability. With 2048 processors, the 16 island multi-master Borg MOEA implementation converged fastest with 100% probability in 410 seconds. With 4096 processors, the 32 island multi-master Borg MOEA implementation dominates, converging with 100% probability in 190 seconds. With 8192 processors, the 16 and 32 island multi-master Borg MOEA implementations perform similarly, converging with 100% probability in 50 and 80 seconds, respectively. Finally, at 16384 processors, the 16 and 32 island multi-master Borg MOEA implementations have nearly identical convergence speeds of approximately 40 seconds. Note at 8192 and 16384 processor counts, the top performing instances of the multi-master Borg MOEA have virtually no random variability. Any given trial of the algorithm is 100% reliable in both solution quality and wall clock time required. This a major benefit for operational use of the algorithm on large parallel architectures where compute hours are often strongly constrained.

From these results, it is clear that the multi-master implementations provide significant improvements in terms of speed and reliability over the master-slave implementations. The master-slave Borg MOEA and ϵ-NSGA-II implementations never converged with 100% probability, regardless of how many processors were available. This failure is attributed to the inefficiency of the master-slave implementations, which quickly become congested trying to receive messages from so many slave nodes (Hadka et al., 2013). Furthermore, the ability of struggling islands to request help from the controller node also is a contributor to the superior performance of the multi-master implementations.

At higher processor counts, inefficiencies due to congestion can also be seen in the 2 and 4 island multi-master implementations. For instance, compare the 2 island multi-master Borg MOEA for the 4096, 8192, and 16384 processor subplots. With 4096 processors, the 2 island multi-master Borg MOEA implementation is performing reasonably well. However, its performance declines significantly with 8192 and 16384 processors. This is a result of each island becoming congested, and it is simply unable to evaluate as many NFE as the implementations with more islands. This shows that selecting a topology appropriate for the processor count is critical. Our simulation-based approach for determining the optimal topology for the multi-master Borg MOEA will be discussed later in Section 8.3.4.

8.3.2 End-of-Run Quality

In the previous section, we analyzed the results in terms of the 90% hypervolume threshold. We fixed the performance threshold and observed the time required to reach this threshold. In this section, we instead fix time and look at the performance of each implementation.

As described in Section 8.2, each implementation was run for 10 minutes. The end-of-run hypervolume is calculated from the approximation set produced by each MOEA after 10 minutes.

Table 8.4 shows the median and standard deviation of the end-of-run hypervolume from all 50 seeds for each implementation. Recall that a hypervolume of 1 is optimal. At 1024 processors, the multi-master Borg MOEA improvement is marginal. The hypervolume increases approximately 2% when switching from the master-slave ϵ-NSGA-II to the 32 island multi-master Borg MOEA. At larger processor counts, the improvement is more significant. With 16384 processors, the 32 island multi-master Borg implementation produces a hypervolume 29% better than master-slave ϵ-NSGA-II. This implies a significant improvement in solution quality when switching to the multi-master Borg MOEA implementation.

Across all topologies, the 16384 processor runs of 32 island multi-master Borg MOEA resulted in the best end-of-run hypervolume. Combined with the speed and reliability results from Section 8.3.1, this shows concretely that the multi-master Borg MOEA with a larger number of islands produces the highest-quality results efficiently and reliably. Furthermore, the results significantly exceed the quality of the master-slave ϵ-NSGA-II and Borg MOEA implementations.

Table 8.4 also provides results from the Kruskal-Wallis and Mann-Whitney U tests. Both tests determine whether differences in the medians of two sampled populations are statistically significant or occurred due to random chance (Sheskin, 2004). The Kruskal-Wallis test is first applied to all medians in the table to determine if there is a statistical difference in the entire table. Since the Kruskal-Wallis test indicated differences were significant, the Mann-Whitney U test is applied to each pair to determine which specific cases are significant. Since the 32 island multi-master Borg MOEA implementation produced the best end-of-run hypervolume, we compare the significance of this result with all other topologies. In Table 8.4, the "significant" column contains a check mark if the end-of-run hypervolume from that row was statistically different from the 32 island multi-master Borg MOEA result with 95% confidence. Additionally, the p-value from the Mann-Whitney U test is shown. With 95% confidence, a p-value ≤ 0.05 rejects the null hypothesis and implies that the results are statistically significant.

These statistical tests show that there is not a statistically significant difference between the 8, 16, and 32 island multi-master Borg MOEA implementations with 16384 processors. However, these three implementations are statistically better than all other runs.

8.3.3 Operator Dynamics

The Borg MOEA bases its selection of search operators on archive membership and recency as discussed in Section 6.2. Adapting its search operators at runtime allows the Borg MOEA to favor operators that contribute more Pareto approximate solutions, leading to faster convergence and diversification. In this section, we explore the operator dynamics on the LRGV problem. The results in this section are based on a single, typical run. We have confirmed that the trends observed in these results are consistent with general trends.

Figure 8.3 shows the operator probabilities from a single run of the master-slave Borg

Table 8.4: Table showing the median and standard deviation of the end-of-run hypervolume results. The Kruskal-Wallis and Mann-Whitney U tests were used to test the statistical significance of the medians. The significant column contains a ✓ if the median from that row is significantly different than the best result, 16384 processor multi-master Borg MOEA (32 islands), with 95% confidence. The row containing the best result is highlighted. The final column contains the corresponding p-value from the Mann-Whitney U test.

Processors	Implementation	Median	Stdev	Significant	p-value
1024	Master-Slave ϵ-NSGA-II	0.88889	0.013124	✓	1.75×10^{-7}
	Master-Slave Borg	0.89146	0.015297	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.89892	0.015105	✓	1.75×10^{-7}
	Multi-Master Borg (4 Islands)	0.89512	0.010933	✓	1.75×10^{-7}
	Multi-Master Borg (8 Islands)	0.90447	0.015395	✓	5.71×10^{-7}
	Multi-Master Borg (16 Islands)	0.90786	0.011394	✓	1.75×10^{-7}
	Multi-Master Borg (32 Islands)	0.90796	0.012429	✓	2.03×10^{-7}
2048	Master-Slave ϵ-NSGA-II	0.89667	0.013536	✓	1.75×10^{-7}
	Multi-Slave Borg	0.88374	0.013262	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.90897	0.014425	✓	3.18×10^{-7}
	Multi-Master Borg (4 Islands)	0.91225	0.013274	✓	3.18×10^{-7}
	Multi-Master Borg (8 Islands)	0.91526	0.014061	✓	2.74×10^{-7}
	Multi-Master Borg (16 Islands)	0.92074	0.015761	✓	3.08×10^{-6}
	Multi-Master Borg (32 Islands)	0.91621	0.012114	✓	2.36×10^{-7}
4096	Master-Slave ϵ-NSGA-II	0.87477	0.014715	✓	1.75×10^{-7}
	Multi-Slave Borg	0.88124	0.013009	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.92561	0.012299	✓	2.36×10^{-7}
	Multi-Master Borg (4 Islands)	0.92572	0.015114	✓	5.27×10^{-6}
	Multi-Master Borg (8 Islands)	0.92695	0.013407	✓	7.82×10^{-6}
	Multi-Master Borg (16 Islands)	0.92601	0.015314	✓	1.49×10^{-5}
	Multi-Master Borg (32 Islands)	0.9332	0.013837	✓	4.01×10^{-5}
8192	Master-Slave ϵ-NSGA-II	0.8163	0.014652	✓	3.61×10^{-7}
	Multi-Slave Borg	0.88813	0.015637	✓	1.75×10^{-7}
	Multi-Master Borg (2 Islands)	0.91815	0.015299	✓	5.71×10^{-7}
	Multi-Master Borg (4 Islands)	0.93421	0.011551	✓	0.000149
	Multi-Master Borg (8 Islands)	0.93698	0.016602	✓	0.010163
	Multi-Master Borg (16 Islands)	0.94167	0.010124	✓	0.005836
	Multi-Master Borg (32 Islands)	0.94194	0.012687	✓	0.025419
16384	Master-Slave ϵ-NSGA-II	0.73672	0.14131	✓	3.39×10^{-6}
	Multi-Slave Borg	0.8907	0.017862	✓	3.39×10^{-6}
	Multi-Master Borg (2 Islands)	0.91252	0.014744	✓	5.05×10^{-6}
	Multi-Master Borg (4 Islands)	0.92989	0.01303	✓	0.000464
	Multi-Master Borg (8 Islands)	0.94489	0.01707		0.21356
	Multi-Master Borg (16 Islands)	0.94534	0.013617		0.53383
	Multi-Master Borg (32 Islands)	0.94814	0.014137		

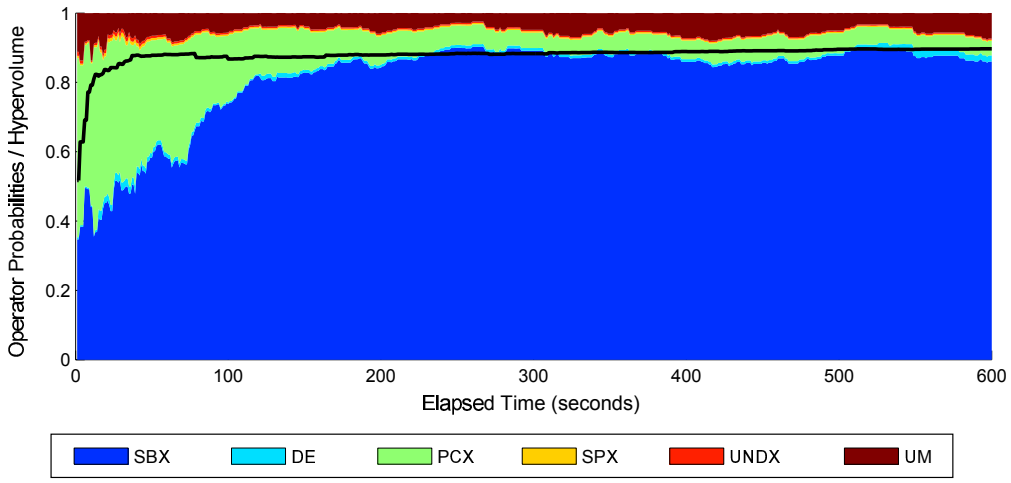


Figure 8.3: The operator probability runtime dynamics from a single run of the master-slave Borg MOEA with 1024 processors. The solid black line traces the hypervolume of the approximation set at each point in time.

MOEA on the LRGV problem with 1024 processors. At each point in time along the x-axis, this plot shows the combination of search operators using the colored regions. Large colored regions corresponding to heavier use of that operator. Additionally, the black solid line traces the hypervolume of the approximation set at each point in time. Although it would be expected that the specific operator probabilities and search dynamics will vary, we have found that they are generally consistent making these results reflective of typical search behavior. The run shown in Figure 8.3 begins with significant use of simulated binary crossover (SBX), parent-centric crossover (PCX), and uniform mutation with probability $1/L$ (UM). These four operators facilitate rapid identification and convergence to the Pareto approximate front. SBX takes over in diversifying along the Pareto front, since SBX with a large distribution index (as with prior studies, this study uses a distribution index of 15) introduces only small perturbations resulting in small, local improvements. Also note that there is no single activated operator, but instead there exists cooperation between several search operators. This cooperation allows the Borg MOEA to combine the qualities of multiple search operators when generating offspring, and can significantly improve the quality of search (Vrugt and Robinson, 2007; Vrugt et al., 2009).

As demonstrated in this example, the use of multiple search operators significantly improves the search dynamics of an MOEA. Membership and recency allow the MOEA to quickly identify the search operators that are beneficial. We also observe that two operators, differential evolution (DE) and unimodal normal distribution crossover (UNDX), had minimal use. While DE and UNDX were not used heavily on the LRGV problem, they have been actively used on other problems (Hadka et al., 2012). Allowing the MOEA to determine the appropriate selection of search operators is a significant advantage when using the Borg

MOEA for real-world complex engineered systems applications.

We can also explore the operator dynamics on the multi-master Borg MOEA. Recall that each island maintains its own operator probabilities, but they can request help from the controller. When receiving help, the island also receives updated operator probabilities that are derived from the global ϵ-dominance archive. Figure 8.4 shows the operator dynamics for a single run of the 16 island multi-master Borg MOEA with 1024 processors. Each of the subplots shows the operator probabilities from a single island. The vertical black lines indicate when the island requests help from the controller. Like Figure 8.3, the solid black line traces the hypervolume of the approximation set at each point in time.

Many islands, as expected, only require help at the end of the run once the initial convergence and diversification is complete. However, we observe that several islands benefit from receiving help earlier in runs. For instance, Island 12 started with significant use of uniform mutation (UM). This selection of operator probabilities was ineffective; the algorithm quickly determined that it was no longer making improvements and immediately asked the controller for help. Upon receiving help, as indicated by the left-most vertical black line, the guidance provided by the controller corrected the operator probabilities to allow search to progress. Thereafter, the algorithm made continuous progress as indicated by the lack of additional help messages until much later in the run. Other islands, such as Island 15, do not require any help during a run.

This example demonstrates how the Borg MOEA can avoid bad initial seeds by relying on the global knowledge gained by running multiple concurrent instances of the Borg MOEA. As we saw with Island 12 in Figure 8.4, an initial bad seed can be quickly detected and corrected without wasting significant computing resources. This contributes to the overall improvement in search quality observed when running the multi-master Borg MOEA with many islands.

We lastly turn to look at the improvement in search quality resulting from the island-based model in the multi-master Borg MOEA. Both Figure 8.3 and Figure 8.4 plot the hypervolume of the approximation set at each point in time with solid black lines. Recall that hypervolumes near 1 indicate high-quality results. The master-slave Borg MOEA search dynamics in Figure 8.3 show that the hypervolume quickly levels off around 0.85 and makes no further improvements. The master-slave Borg MOEA is simply unable to attain high-quality results. However, by running multiple islands and sharing solutions and operator probabilities between islands as done in the multi-master Borg MOEA, hypervolume is increased substantially. Figure 8.4 shows this effect. While individual islands tend to converge slower than the master-slave run in Figure 8.3, they attain substantially better hypervolume results later in the run.

8.3.4 Parallel Efficiency and Speedup

Finally, we explore the parallel efficiency and speedup of the various master-slave and multi-master Borg MOEA configurations explored in this study. Since each implementation was run for a fixed wallclock time (10 minutes), efficiency is based on the total NFE in each run. Thus, if NFE_S is the total NFE for a serial run and NFE_P is the total NFE for a parallel

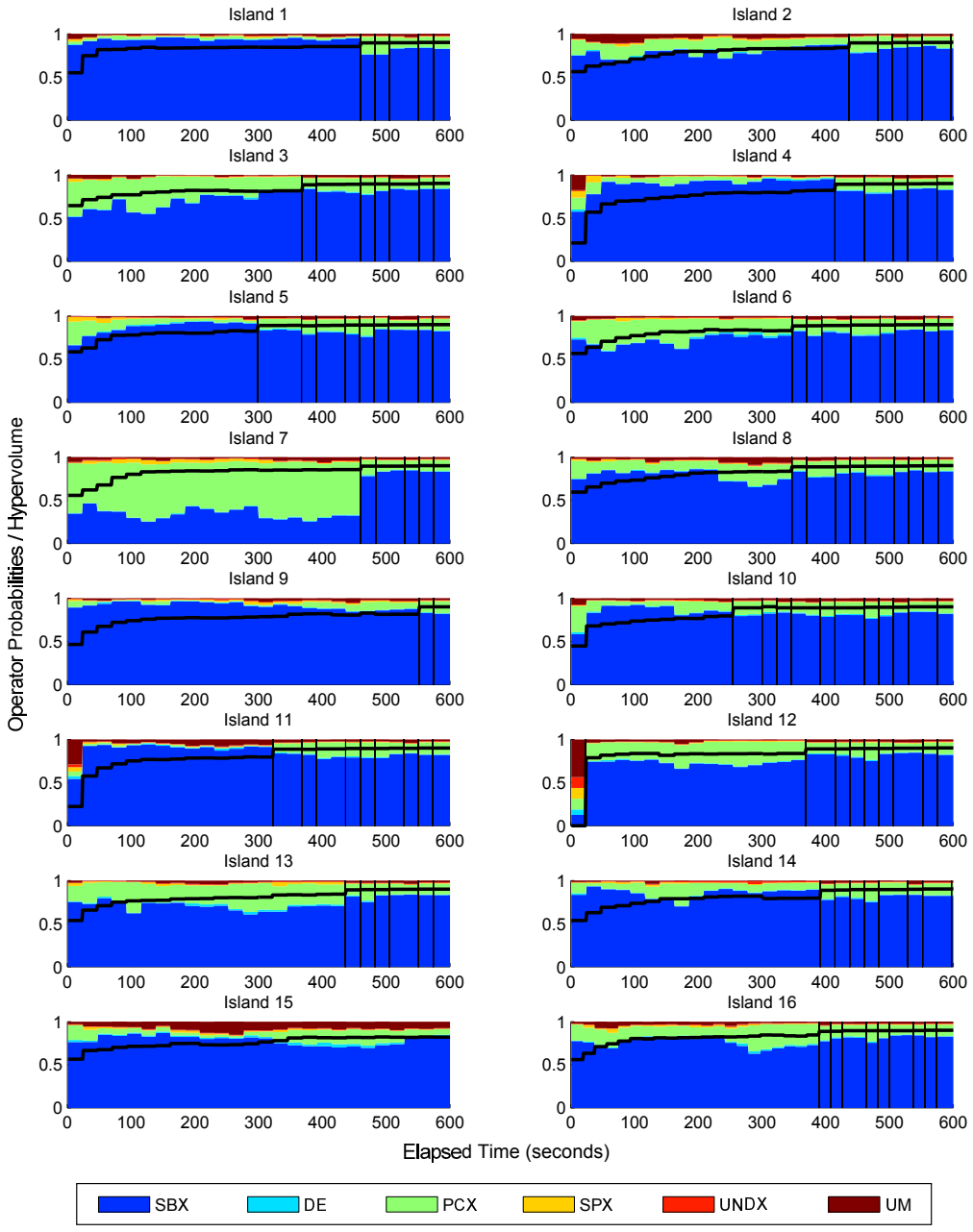


Figure 8.4: The operator probability runtime dynamics from a single run of the 16 island multi-master Borg MOEA with 1024 processors. Each subplot shows the operator probabilities for an island. The vertical black lines indicate when the island requested help from the controller. Like Figure 8.3, the solid black line traces the hypervolume of the approximation set at each point in time.

Table 8.5: Table showing the median NFE expended by each implementation and the parallel efficiency.

Processors	Implementation	Total NFE	Efficiency	Predicted Efficiency
1024	Multi-Slave Borg	4293080	0.978	0.98
	Multi-Master Borg (2 Islands)	4301767	0.98	0.99
	Multi-Master Borg (4 Islands)	4291951	0.978	0.99
	Multi-Master Borg (8 Islands)	4277744	0.975	0.98
	Multi-Master Borg (16 Islands)	4242323	0.967	0.97
	Multi-Master Borg (32 Islands)	4166046	0.949	0.96
2048	Multi-Slave Borg	7755607	0.884	0.91
	Multi-Master Borg (2 Islands)	8610209	0.981	0.97
	Multi-Master Borg (4 Islands)	8609865	0.981	0.98
	Multi-Master Borg (8 Islands)	8588290	0.979	0.98
	Multi-Master Borg (16 Islands)	8552526	0.975	0.97
	Multi-Master Borg (32 Islands)	8478679	0.966	0.97
4096	Multi-Slave Borg	7681163	0.438	0.47
	Multi-Master Borg (2 Islands)	16496460	0.94	0.91
	Multi-Master Borg (4 Islands)	17174236	0.979	0.97
	Multi-Master Borg (8 Islands)	17207637	0.98	0.97
	Multi-Master Borg (16 Islands)	17142685	0.977	0.97
	Multi-Master Borg (32 Islands)	17129074	0.976	0.96
8192	Multi-Slave Borg	7160437	0.204	0.23
	Multi-Master Borg (2 Islands)	17057671	0.486	0.46
	Multi-Master Borg (4 Islands)	32469898	0.925	0.92
	Multi-Master Borg (8 Islands)	34009570	0.969	0.97
	Multi-Master Borg (16 Islands)	34139711	0.973	0.98
	Multi-Master Borg (32 Islands)	34121055	0.972	0.98
16384	Multi-Slave Borg	4470551	0.064	0.08
	Multi-Master Borg (2 Islands)	14385033	0.205	0.23
	Multi-Master Borg (4 Islands)	32373010	0.461	0.47
	Multi-Master Borg (8 Islands)	64639837	0.921	0.91
	Multi-Master Borg (16 Islands)	67101524	0.956	0.96
	Multi-Master Borg (32 Islands)	67661785	0.964	0.97

run with P processors, efficiency is calculated by

$$\text{efficiency} = \frac{\text{NFE}_P}{P \cdot \text{NFE}_S}. \tag{8.6}$$

The total NFE of the serial algorithm running for 10 minutes is $\text{NFE}_S = 4285$. Table 8.5 shows the total NFE expended by each parallel implementation and the calculated efficiency.

With only 1024 processors, all of the configurations have very high efficiency. As expected, as the number of islands increases, the efficiency drops slightly due to the overhead introduced by having additional master nodes, the controller node, and the additional communication between these nodes.

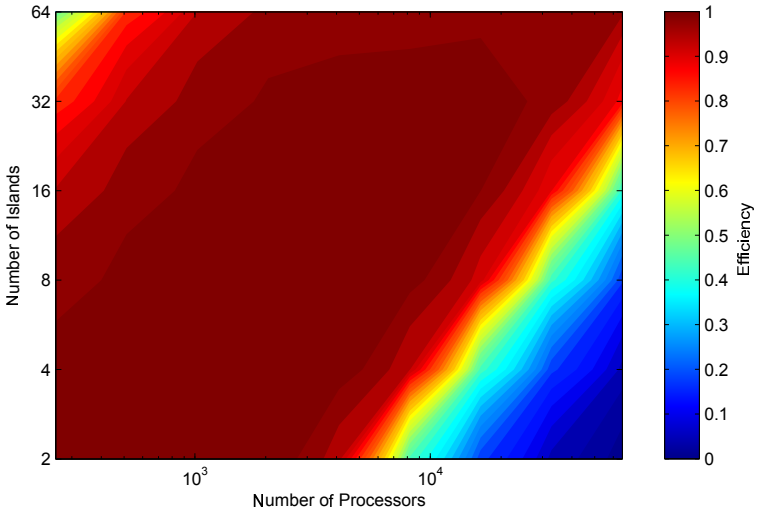


Figure 8.5: Predicted efficiency for the multi-master Borg MOEA on the LRGV problem from 1024 up to 65536 processors.

When the number of processors increases beyond 1024, the efficiency of the master-slave Borg MOEA rapidly declines. With 16384 processors, the master-slave Borg MOEA is running with an efficiency of 0.064. At this point, the increased overhead and communication burden overloads the single master node and reduces the overall NFE. Increasing the number of islands reduces the workload on individual master nodes, spreading the NFE across multiple islands. Looking at the 16384 processor case in Table 8.5, switching from the master-slave to a 2 island multi-master configuration increases the efficiency from 0.064 to 0.205. Increasing the number of islands improves the efficiency further, reaching an efficiency of 0.964 with 32 islands.

In Chapter 7, we developed a discrete event simulation model for accurately predicting the efficiency of the master-slave and multi-master Borg MOEA. Table 8.5 shows the actual and the predicted efficiency from this model for the LRGV problem. Timing collected from the LRGV runs determined the inputs to the simulation model. These inputs included estimates for the algorithm overhead, $T_A = 0.000105$ seconds, the communication overhead, $T_C = 0.000006$ seconds, and objective function evaluation time, $T_F = 0.14$ seconds. All of these timings were collected on TACC Ranger. From Table 8.5, we see that the simulation model can very accurately predict the parallel efficiency of the multi-master Borg MOEA.

We expect the multi-master Borg MOEA to be able to efficiently scale to very large processor counts by increasing the number of islands as needed to remain efficient. Using the simulation model, we can predict the efficiency of the multi-master Borg MOEA at larger processor counts. Figure 8.5 shows the predicted efficiency for the LRGV problem. Note the linear relationship between the number of processors and the number of islands. To maintain high efficiency, doubling the number of processors requires the number of islands to

double. This maintains a fixed number of processors per island, which is chosen to yield the maximum efficiency. We can use this simulation model to determine the optimal topology for maximizing efficiency.

Maximizing efficiency will increase NFE, but this does not necessarily correspond to increased search quality. It is also necessary to consider how parallelization improves overall search quality. Figure 8.6 shows the comparative speedup attained when switching from the master-slave to the multi-master Borg MOEA. Each subplot corresponds to a different processor count. The lines within each subplot trace the speedup of that implementation. The baseline is the master-slave Borg MOEA. Results are averaged over 50 random seed trials. The speedup measures how many times faster (or slower) the multi-master Borg MOEA is in attaining the same hypervolume. For example, if the master-slave Borg MOEA reached a hypervolume of 0.8 in 300 seconds, and the multi-master Borg MOEA reached the same hypervolume in 150 seconds, it would show a speedup of 2. Since the master-slave is the baseline, it appears as a flat line with a speedup of 1. Note that these speedup measurements are provided between runs with the same processor count — the computing power is fixed. Thus, any speedup observed is a result of the improved convergence and diversity of a given implementation of the parallel Borg MOEA, and is not a result of more computing power.

With 1024 processors, we see that at low hypervolume thresholds, the multi-master Borg MOEA implementations have lower convergence speeds than the maser-slave Borg MOEA. Only as we increase the hypervolume threshold do the multi-master Borg MOEA implementations begin to converge faster. The master-slave Borg MOEA converges very fast, but it is limited to attaining lower hypervolume than the multi-master Borg MOEA. Note in Figure 8.6 given that the master-slave Borg MOEA baseline never attains the highest levels of hypervolume, the multi-master Borg MOEA speedup results are conservative. At the largest tested processor count, 16384, we see that the 16 and 32 island multi-master Borg MOEA runs reach a speedup of $10-18$ times faster than the master-slave Borg MOEA. This means these multi-master runs are converging in 1/10th the wallclock time as the master-slave Borg MOEA, even though the multi-slave and multi-master are given the same number of processors. This speedup is therefore a result of algorithmic improvements in the multi-master paradigm, allowing the algorithm to capture the same solution quality in less time. This combined with the global restarts and guidance provided by the controller help improve the speed, effectiveness, and reliability of the multi-master Borg MOEA.

8.4 Conclusion

The Borg MOEA was originally introduced to solve many-objective, multi-modal, non-separable engineering problems. The success of the Borg MOEA has been demonstrated in several studies (Hadka and Reed, 2012a,b; Hadka et al., 2012; Reed et al., 2012). Application of the Borg MOEA is limited by its serial implementation, which is unable to rapidly solve large-scale problems with expensive objective function evaluations.

To address this limitation, this study developed two parallel versions of the Borg MOEA. The master-slave Borg MOEA runs a parallelized version of the serial Borg MOEA where

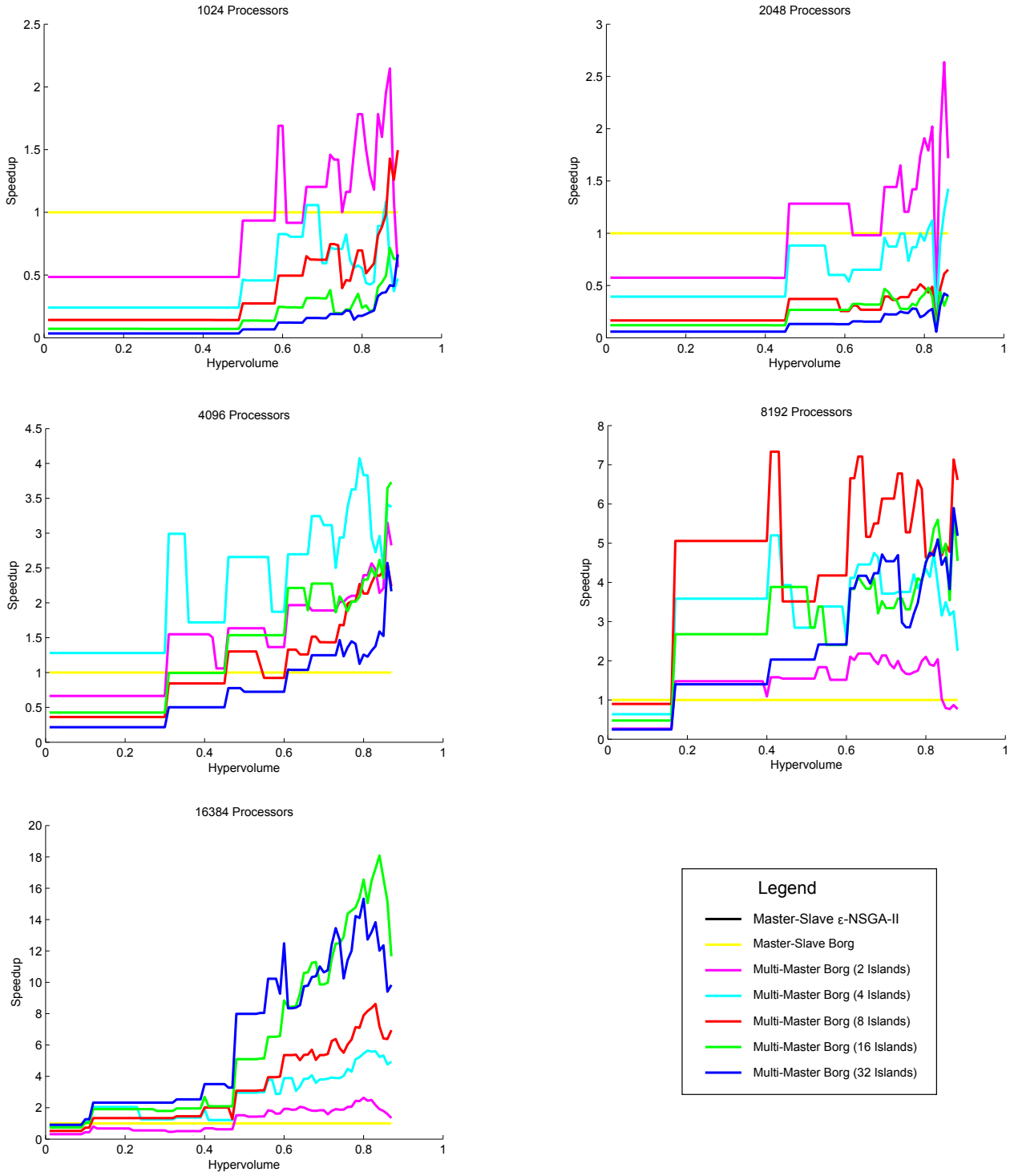


Figure 8.6: Hypervolume speedup of the multi-master Borg MOEA implementations compared to the baseline master-slave Borg MOEA. These results are averaged over the 50 random seed trials.

objective function evaluations are performed in parallel. This provides direct speedup, but is limited by inefficiencies due to the communication overhead that limits its ability to attain very high levels of performance. The multi-master Borg MOEA is a hierarchical extension where two or more islands run instances of the master-slave Borg MOEA in parallel. Additionally, a global controller node maintains the global search state of the algorithm and provides guidance to masters when they preconverge. This guidance extends the restart mechanism and the adaptive selection of search operators of the serial Borg MOEA, allowing for global restarts and sharing of the global search state.

Applying these parallel implementations of the Borg MOEA to a risk-based water supply portfolio planning problem, we observed that the master-slave and multi-master Borg MOEA produced high-quality solutions when compared to another state-of-the-art parallel MOEA, ϵ-NSGA-II. The multi-master Borg MOEA with 32 islands produced the highest-quality results. This is attributed to the ability of the multi-master implementation to quickly detect preconvergence in islands and provide guidance in the form of the global ϵ-dominance archive and global operator probabilities.

The efficiency, reliability, and search quality of the multi-master Borg MOEA have been demonstrated running on up to 16384 processors with over 95% efficiency. We contribute an accurate discrete event simulation of the multi-master Borg MOEA's parallel efficiency that shows the algorithm has the strong potential for use on emerging Petascale and planned Exascale computing architectures (> 100000 processors). The ability to scale efficiently to high processor counts makes the Borg MOEA a viable tool for solving extremely large-scale, complex engineering problems. For the LRGV problem explored in this study, the 32 island multi-master Borg MOEA solved the problem with the highest-quality results in 10 minutes using 16384 processors. If running in serial, this would require over 109 days of computation. This opens the possibility for solving such complex engineered systems effectively while providing decision-makers with the ability to rapidly evaluate their tradeoffs, formulations, and potential design solutions.

Chapter 9

Conclusions, Contributions, and Future Work

Multiobjective evolutionary algorithms (MOEAs) are changing the way we think about engineering optimization problems. Traditionally, decision-makers would formulate their problem based on *a priori* domain knowledge. In doing so, the decision-maker is introducing decision bias into their problem. Use of aggregate or lower-dimensional formulations of a problem may cause decision-makers to inadvertently ignore tradeoffs that would otherwise influence their decision preferences (i.e., cognitive myopia) (Hogarth, 1981). Additionally, highly constrained and aggregated formulations often yield alternatives that are strongly influenced by the decision-maker's preconceptions of a system, strongly limiting the discovery of tradeoffs and design alternatives (i.e., cognitive hysteresis) (Gettys and Fisher, 1979). Both cognitive myopia and cognitive hysteresis negatively impact the outcome from optimization.

To avoid these downfalls, Woodruff et al. (2013) proposes a many-objective visual analytics (MOVA) framework wherein problem formulation, many-objective optimization, negotiated design selection, and interactive visualization work together seamlessly, allowing information to feedback between each stage and potentially alter the design. This approach necessitates the exploration of higher-dimensional, many-objective spaces. It is therefore necessary to employ many-objective search tools that can effectively search the high-dimensional problem formulations, capture the complex tradeoffs between the objectives, and allow fluid feedback between the optimization process, design selection, interactive visualization, and problem (re)formulation. There exist a number of challenges in this regard, particularly when solving complex engineered systems. Complex engineered systems often feature challenging problem properties, including (1) many-objective formulations, (2) multi-modality (or false optima), (3) nonlinearity, (4) discreteness, (5) severe constraints, (6) stochastic objectives, and (7) non-separability (also called epistasis) (Reed et al., 2012). These properties prohibit the use of traditional optimization techniques and often require the use of metaheuristics like MOEAs.

In this dissertation, we developed an adaptive, many-objective optimization tool called the Borg MOEA. The Borg MOEA features a number of adaptive mechanisms allowing the algorithm to tailor itself to local search conditions encountered during optimization.

It can auto-adapt its use of multiple search operators conditional on their performance, it tracks all Pareto approximate solutions discovered during search, and it triggers adaptive restarts to escape local optima and avoid preconvergence. This allows the Borg MOEA to seamlessly handle many-objective formulations, multi-modality, nonlinearity, discreteness, severe constraints, stochastic objectives, and non-separability. The effectiveness of the Borg MOEA has been demonstrated on a number of analytical test problems and several real-world applications throughout this dissertation. In nearly all test cases, the Borg MOEA meets or exceeds the performance of other contemporary MOEAs.

To assist in the experimental comparison of algorithms, we developed a rigorous testing framework to compare the relative performance of MOEAs. To this end, we proposed several metrics for comparing different MOEAs based on their efficiency, reliability, and controllability. This framework also includes a diagnostic tool to assess the relative importance of an MOEA's parameters. It is well-documented that many MOEAs are sensitive to their parameterizations, and the optimal parameterization often changes across problem domains and even problems within a single domain (Purshouse and Fleming, 2003, 2007). Applying this diagnostic tool to the state-of-the-art MOEAs, we identified that most MOEAs are extremely sensitive to small parameter perturbations. The Borg MOEA, because of its auto-adaptiveness, remains insensitive to all parameters except for NFE. This trend has been observed on over 30 analytical test problems and several real-world applications.

Since the Borg MOEA's performance has been shown to be controllable using NFE, we hypothesized it will benefit heavily from parallelization. Therefore, we developed two parallel versions of the Borg MOEA. The first version, the master-slave Borg MOEA, is a straightforward parallelization of the Borg MOEA. We observed that the master node quickly becomes a bottleneck as the number of processors increases, and consequently we developed a hierarchical extension called the multi-master Borg MOEA. The multi-master Borg MOEA can scale to tens of thousands of processors and maintain high efficiency. The multi-master Borg MOEA not only increases NFE due to increased efficiency but also results in an overall improvement in speedup and search quality due to the dynamics introduced by the controller node. The controller is tasked with monitoring search progress within each island / master and will trigger global restarts and provide guidance in the form of Pareto approximate solutions and operator preferences.

As the Borg MOEA is insensitive to parameterization, we need not be concerned about the parameterization of each island in the multi-master Borg MOEA. However, the topology is important for improving efficiency and search quality. To assist the design of the topology, we developed a discrete event simulation model to predict the optimal processor count per island in order to maximize efficiency. Using this model, we can configure the multi-master to maximize its utilization of the underlying parallel architecture.

This dissertation culminates with the application of the Borg MOEA to a severely constrained, many-objective complex engineered system: a risk-based water supply portfolio planning problem. Not only does this problem exhibit all of the challenging problem properties, its expensive function evaluation times necessitate large-scale parallelization. The multi-master Borg MOEA scaled efficiently to 16384 processors on the TACC Ranger su-

percomputer to solve this problem efficiently and reliably, producing the highest-quality solutions for this problem to date.

The Borg MOEA is changing the way we optimize complex engineered systems. It is highly-controllable, efficient, scalable, and consistently produces high-quality solutions on complex, many-objective problems. As a result, the Borg MOEA can be integrated within the MOVA framework (Woodruff et al., 2013) to provide rapid feedback to the decision-makers. Cognitive myopia and cognitive hysteresis can be eliminated through the optimization of high-dimensional formulations of complex engineered systems, leading to a revolution in decision-making.

9.1 Contributions

This dissertation developed the serial, master-slave, and multi-master variants of the Borg MOEA. Throughout this work, we have provided many key contributions to the field of parallel computing, evolutionary computation, and operations research. This sections summarizes the contributions of this dissertation.

9.1.1 Technical Contributions

The following technical contributions resulted from the work in this dissertation. These technical contributions represent novel research that contributed to the fields of parallel computing, evolutionary computation, and operations research.

- The Borg MOEA is the first MOEA to combine ϵ-dominance, adaptive population sizing, and auto-adaptive multioperator recombination to produce a highly adaptive and scalable MOEA. The Borg MOEA has been shown to be superior to existing tools using numerous test problems (see Chapters 3 and 4) and several real-world case studies (see Chapters 5 and 8).

- Detailed investigation of the auto-adaptive multioperator recombination selection probabilities in the Borg MOEA provides strong insights into the dynamics of operator / topology interaction. In other words, researchers can use the operator selection probabilities to infer information about a problem's topology, including identifying problems favoring mean-centric operators (simple landscape), parent-centric operators (multimodality), or rotationally-invariant operators (high degree of epistasis).

- The diagnostic framework presented in Chapter 4 is the first framework to statistically assess an MOEA's search quality, reliability, controllability, and efficiency. In particular, controllability and efficiency are two novel metrics developed for this work.

- The diagnostic framework also applies Sobol' sensitivity analysis to ascertain the search controls and failure modes of MOEAs by investigating the effects and interactions between algorithm parameters. This provided the first detailed understand of the key factors that control an MOEA's performance.

- In Chapter 5, we demonstrated that the Borg MOEA outperforms all tested contemporary MOEAs on a severely constrained, many-objective product family design problem. Additionally, Chapter 5 applies the diagnostic framework from Chapter 4 to this product family design problem, confirming that the Borg MOEA is highly controllable on a real-world application.

- In Reed et al. (2012), we demonstrated that the Borg MOEA is a top contender on three real-world, complex water engineering applications. These applications range from identifying optimal water planning portfolios (the LRGV problem) to flood forecasting. This demonstrates that the Borg MOEA is both reliable and highly competitive across a range of problem domains.

- The master-slave and multi-master Borg MOEA developed in Chapter 6 provide highly efficient and reliable many-objective optimization with the ability to scale to tens of thousands of processors. Complex engineered systems can be solved efficiently, providing rapid feedback to decision-makers. As a result, the Borg MOEA is a candidate optimization tool within the many-objective visual analytics (MOVA) framework developed by Woodruff et al. (2013). The MOVA framework aims to provide fluid feedback between the problem formulation, many-objective search, and visualization.

- In Chapter 7, we develop the first theoretical scalability models for parallel, asynchronous MOEAs. We identify the limits on scalability, processor count lower and upper bounds, speedup, and efficiency. This work also demonstrated the ability of simulation models to accurately model complex, parallel architectures.

- In Chapter 8 we apply the parallel Borg MOEA implementations to a complex engineered system: a risk-based water supply portfolio planning problem. In doing so, we scaled the algorithms to 16384 processors with over 95% efficiency. This problem can be solved with high-reliability in under a minute. Without parallelization, it would take over 114 days to execute a similar study.

- The complex engineered system optimized in Chapter 8 was used to explore the adaptive parameterization in the multi-master Borg MOEA. The dynamic global restarts enabled by the controller node and its guidance is shown to fundamentally improve the speed and quality of search.

9.1.2 Peer-Reviewed Journal Articles

The following peer-reviewed journal articles resulted from the work presented in this dissertation. This includes manuscripts currently being prepared for submission to top journals in the fields of parallel computing, evolutionary computation, and civil engineering.

- Hadka, D., et al. *Large-scale Parallelization of the Borg MOEA for Addressing the Design of Complex Engineered Systems.* Evolutionary Computation, In-Preparation.

- Reed, P., et al. *Evolutionary Multiobjective Optimization in Water Resources: The Past, Present & Future.* (Editor Invited Submission to the 35th Anniversary Special Issue), Advances in Water Resources, 2012.

- Hadka, D. and Reed, P. *Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization.* Evolutionary Computation, 20(3):423–452, 2012.

- Hadka, D. and Reed, P. *Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework.* Evolutionary Computation, 2012.

9.1.3 Presentations at Conferences and Invited Talks

- Hadka, D., et al. *Scalability Analysis of the Multi-Master Borg Multiobjective Evolutionary Algorithm.* Supercomputing, In-Preparation.

- Hadka, D., et al. *Scalability Analysis of the Asynchronous, Master-Slave Borg Multiobjective Evolutionary Algorithm.* 27th International Parallel & Distributed Processing Symposium (IPDPS), Nature Inspired Distributed Computing Workship (NIDISC), Boston, MA, 20-24 May 2013 (To Appear).

- Kasprzyk, J. et al. *Diagnostic Evaluation of Many Objective Search for Water Supply Portfolio Planning.* World Environmental and Water Resources Congress, Cincinnati, Ohio, 19-23 May 2013 (To Appear).

- Hadka, D., et al. *Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems.* INFORMS, Phoenix, Arizona, 16 October 2012.

- Kollat, J., et al. *Evolutionary Multiobjective Optimization in Water Resources: The Past, Present, and Future.* INFORMS, Phoenix, Arizona, 16 October 2012.

- Woodruff, M., et al. *Auto-Adaptive Search Capabilities of the New Borg MOEA: A Detailed Comparison on Product Family Design Problems.* 12th AIAA Aviation Technology, Integration, and operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Indianapolis, Indiana, 17 September 2012.

- Reed, P., et al. *Evolutionary Multiobjective Optimization in Water Resources: The Past, Present, and Future.* International Environmental Modelling and Software Society, Leipzig, Germany, July 2012.

- Hadka, D., et al. *Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems.* WCCI 2012 World Congress on Computational Intelligence, Congress on Evolutionary Computation, Brisbane, Australia, 10-15 June 2012.

- Reed, P., et al. *A Diagnostic Assessment of Evolutionary Multiobjective Optimization for Water Resources Systems.* European Geosciences Union (EGU) General Assembly, Vienna, Austria, 27 April 2012.

- Reed, P. and Hadka, D. *Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization.* INFORMS 2011 Annual Meeting, Charlotte, NC, 14 November 2011.

- Reed, P. and Hadka, D. *Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization.* ASCE World Water and Environmental Resources Congress, Palm Springs, CA, May 2011.

- Presented at the University of Exeter, the University of Sheffield and the Aerospace Corporation by Dr. Patrick Reed. These talks focused on the work presented in Chapter 4.

9.1.4 Patents

The motivation for securing patent protection revolves around corporate interest in such technologies. Many corporations are actively deploying MOEAs to solve real-world applications, and such patent protection secures the intellectual property and marketability of the Borg MOEA. The innovations underlying the Borg MOEA are patent pending with the United States Patent and Trademark Office (USPTO) under patent application 13/356,391, filed 23 January 2012. Provisional patent application 61/766,607 for the multi-master approach to parallelizing the Borg MOEA was filed on 19 February 2013.

9.1.5 Software

Most of the software developed in this dissertation is available freely for non-commercial research.

- The serial Borg MOEA from Chapter 3 is available at `http://www.borgmoea.org`, licensed under the Pennsylvania State University Research and Educational Use License. This software is written in ANSI C. At the time of writing, 42 academic users from 16 universities are currently using the serial Borg MOEA.

- The MOEA Framework is an open source Java library that supports the design, experimentation, and analysis of over 25 evolutionary and nature-inspired MOEAs. This library also contains the diagnostic framework developed in Chapter 4. The MOEA Framework is available at `http://www.moeaframework.org`, licensed under the GNU Lesser General Public License. At the time of writing, the MOEA Framework has been downloaded 3500 times by individuals from 72 countries since its initial release in December 2011, currently averaging about $60 - 70$ downloads a week.

- The General Aviation Aircraft (GAA) problem tested in Chapter 5 is licensed under the open source MIT license at `http://www.coe.psu.edu/water/index.php/Benchmark_Data`. This includes C and Java implementations compatible with the Borg MOEA and the MOEA Framework, respectively. This code was originally developed by Timothy Simpson and Ruchit Shah (Shah et al., 2011) but was modified as part of this dissertation.
- A modified version of the master-slave Borg MOEA from Chapter 6 was delivered to the Colorado Springs Utilities for use in optimizing their complex water resources system.
- The master-slave and multi-master Borg MOEA implementations are available by request. Please contact Patrick Reed at `preed@engr.psu.edu` for details.

9.2 Future Work

This section proposes several directions for future work to extend and improve the Borg MOEA.

Real-World Case Studies
This dissertation presents two real-world case studies in Chapters 5 and 8. Comparative studies that explore the effectiveness and behavior of the Borg MOEA, both serial and parallel variants, will help strengthen the results presented in this dissertation. Of interest are problems of varying size (both in the number of decision variables and objectives), complexity, and function evaluation time.

Adaptive Topology
In Chapter 7, we devised a strategy for configuring the parallel topology to maximize the parallel efficiency and hypervolume speedup of the parallel Borg MOEA. This requires the user to first collect timing data from the problem (function evaluation time, algorithm overhead) and the computer architecture (communication costs). One can consider if the parallel topology of the Borg MOEA can be adapted on-line, allowing the Borg MOEA to alter its topology to maximize performance.

Petascale Application
While the parallel Borg MOEA was applied on up to 16384 processors, the discrete event simulation results indicate it will remain efficient as it scales on Petascale systems (i.e., architectures with greater than 100000 processors). Applying the Borg MOEA on Petascale systems will validate the experimental and analytical results presented in this dissertation, and also demonstrate the power of the multi-master Borg MOEA.

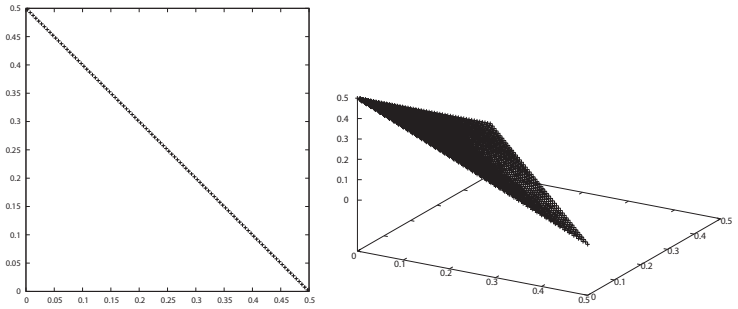
Wide-Area-Network (WAN) Application
Unlike local-area-networks (LANs), wide-area-networks (WANs) are networks spread across large geographic regions. As a result, WANs experience large communication latencies that would render the parallel Borg MOEA extremely inefficient. WANs also tend to have heterogeneous hardware, which impacts function evaluation times and algorithmic overhead. We propose investigating the application of the parallel Borg MOEA to WANs. This would eliminate the need to request allocation time on large-scale computing architectures, and would instead run on an ad-hoc parallel system formed by a distributed network of donated computing time. Significant changes to how the parallel Borg MOEA is distributed across the network would be required. For instance, to reduce communication costs, the algorithm can partition the networked computers based on their communication costs into faster, local subnets.

Alternative Representations
Throughout this dissertation, we explored the behavior of the Borg MOEA on real-valued (and integer-valued) problems using six real-valued search operators. MOEAs have also been successful on problems with discrete representations, such as binary strings and permutations. We propose exploring the set of discrete search operators to determine what combinations work effectively within the Borg framework. These operators should be selected to diversify the offspring distributions, which allows the Borg MOEA to auto-adapt its use of those operators that produce favorable offspring.

Appendix A

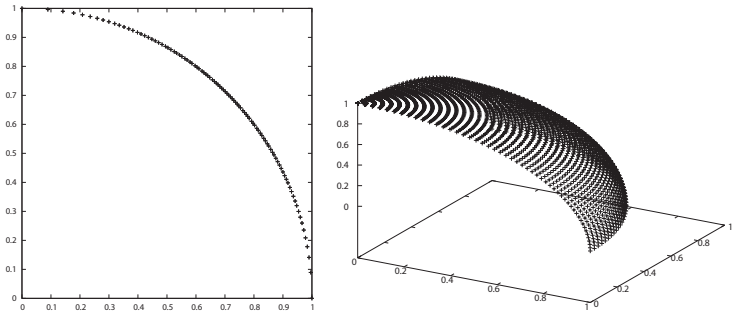
Multiobjective Problems

This appendix provides examples of the reference sets for select test problems from the DTLZ, CEC 2009, and WFG test problem suites (Deb et al., 2002b; Zhang et al., 2009b; Huband et al., 2006). Many of these problems are designed to be scalable to any objective dimension. Only the 2 and 3 objective cases are shown for such problems.



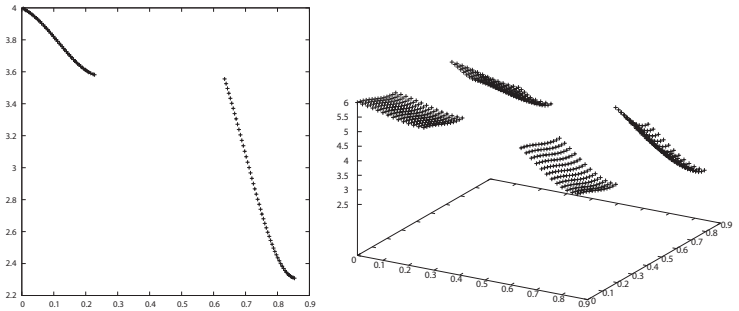
(a) 2D DTLZ1 (b) 3D DTLZ1

Figure A.1: Reference sets for the DTLZ1 test problem with 2 and 3 objectives. UF12 from the CEC 2009 competition is a 5 objective rotated variant of DTLZ1.



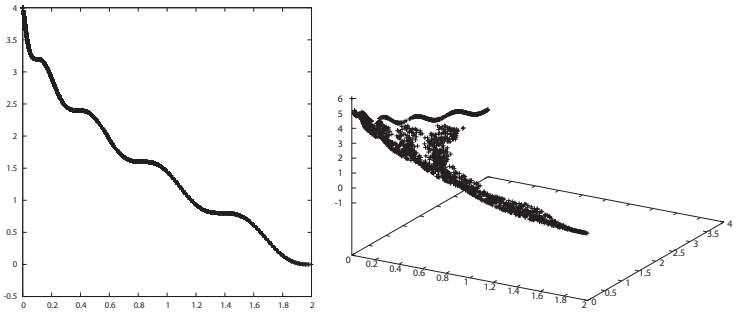
(a) 2D DTLZ2-DTLZ4 (b) 3D DTLZ2-DTLZ4

Figure A.2: Reference sets for the DTLZ2, DTLZ3, and DTLZ4 test problems with 2 and 3 objectives. While these three problems share the same reference set, their objective definitions differ dramatically. For instance, DTLZ3 is considerably more difficult than DTLZ2 due to the addition of multi-modality. UF11 from the CEC 2009 competition is a 5 objective rotated variant of DTLZ2.



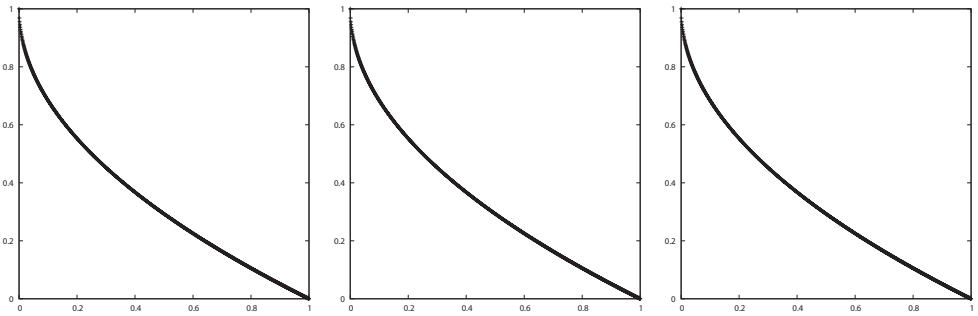
(a) 2D DTLZ7 (b) 3D DTLZ7

Figure A.3: Reference sets for the DTLZ7 test problem with 2 and 3 objectives.

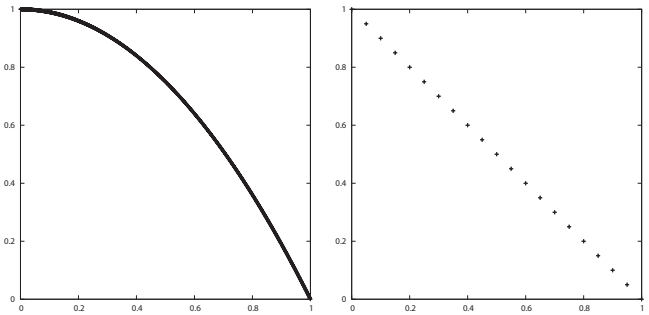


(a) 2D WFG1 (b) 3D WFG1

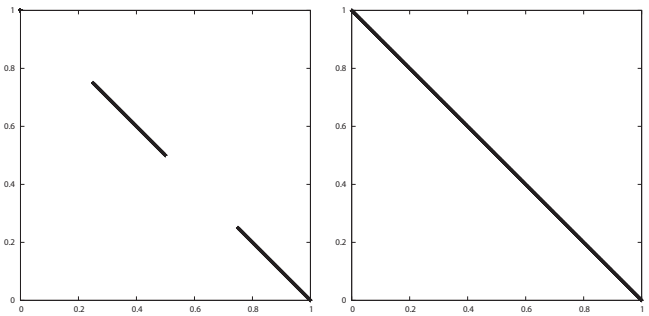
Figure A.4: Reference sets for the WFG1 problem for 2 and 3 objectives. UF13 from the CEC 2009 competition is the 5 objective variant of WFG1.



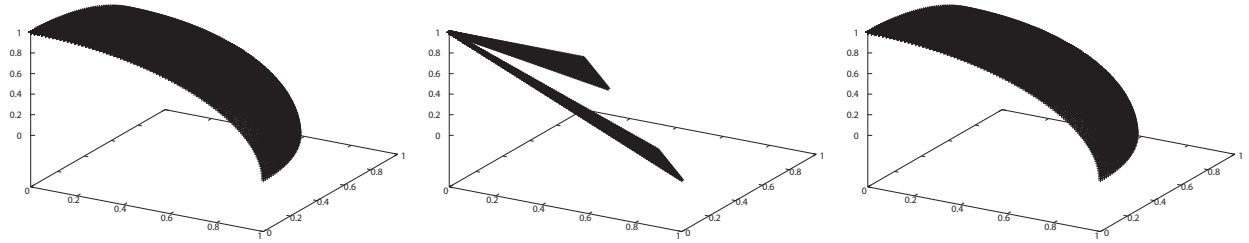
(a) UF1 (b) UF2 (c) UF3



(d) UF4 (e) UF5



(f) UF6 (g) UF7



(h) UF8 (i) UF9 (j) UF10

Figure A.5: Reference sets for the unconstrained problems from the CEC 2009 competition.

Appendix B

Sobol's Global Variance Decomposition

Using the notation and terminology of Saltelli et al. (2008), given a square-integrable function f transforming inputs $X_1, X_2, \ldots, X_n$ into output Y,

$$Y = f(X_1, X_2, \ldots, X_n), \tag{B.1}$$

the global variance decomposition technique proposed by I. M. Sobol' considers the following expansion of f into terms of increasing dimension:

$$f = f_0 + \sum_i f_i + \sum_{i<j} f_{ij} + \sum_{i<j<k} f_{ijk} + \cdots + f_{ijk\ldots n}, \tag{B.2}$$

where each individual term is a function only over the inputs in its index (Saltelli et al., 2008; Archier et al., 1997). For example, $f_i = f_i(X_i)$ and $f_{ij} = f_{ij}(X_i, X_j)$. Sobol' proved that the individual terms can be computed using conditional expectations, such as

$$f_0 = E(Y), \tag{B.3}$$

$$f_i = E(Y|X_i) - f_0, \tag{B.4}$$

$$f_{ij} = E(Y|X_i, X_j) - f_i - f_j - f_0. \tag{B.5}$$

If the output Y is sensitive to input X_i, then the conditional expectation $E(Y|X_i)$ has a large variance across the values of X_i. Hence, the variance of the conditional expectation is a measure of sensitivity. The first-order effects are calculated by

$$S_i = \frac{V[f_i(X_i)]}{V[Y]} = \frac{V[E(Y|X_i)]}{V[Y]}. \tag{B.6}$$

The second-order effects are calculated with

$$S_{ij} = \frac{V[f_{ij}(X_i, X_j)]}{V[Y]} \tag{B.7}$$

$$= \frac{V[E(Y|X_i, X_j)]}{V[Y]} - S_i - S_j. \tag{B.8}$$

An important consequence of Sobol's work is the computation of total-order effects. The total effects caused by input X_i is the sum of the first-order effect S_i and all higher-order effects influenced by X_i. Thus, total-order effects are calculated by

$$S_i^T = 1 - \frac{V[E(Y|X_{\sim i})]}{V[Y]}, \tag{B.9}$$

where $X_{\sim i}$ represents all the inputs excluding X_i. Saltelli et al. (2008) developed the Monte Carlo technique for efficiently computing the first-, second-, and total-order effects used in this study. To validate the sensitivity results, the bootstrap technique called the *moment method* produces symmetric 95% confidence intervals, as described in Archier et al. (1997) and Tang et al. (2007). The moment method provides more reliable results with smaller resampling sizes so long as the distribution is not skewed left or right (Archier et al., 1997). We chose a resampling size of 2000 since it is both recommended in the literature and experimentally robust (Tang et al., 2007). Interested readers should refer to the cited materials for additional details.

Appendix C

Asynchronous MOEA SimPy Model

The SimPy model for the asynchronous MOEA is reproduced below.

```
from SimPy.Simulation import *
from random import *
from math import *
from sys import *

class State:
  def __init__(self, master, slave, masterMonitor, slaveMonitor, P
      , maxNFE, Ta, Tc, Tf):
    self.master = master
    self.slave = slave
    self.masterMonitor = masterMonitor
    self.slaveMonitor = slaveMonitor
    self.NFE = 0
    self.maxNFE = maxNFE
    self.P = P
    self.Ta = Ta
    self.Tc = Tc
    self.Tf = Tf
    self.TaStdev = 0.1*Ta
    self.TcStdev = 0.1*Tc
    self.TfStdev = 0.1*Tf
    self.start = now()

  def waitTime(self):
    return self.masterMonitor.mean()

  def duration(self):
    return now()-self.start
```

```
28
29    def speedup(self):
30      return (self.maxNFE * self.Tf) / self.duration()
31
32    def efficiency(self):
33      return self.speedup() / self.P
34
35  class Solution(Process):
36    def generate(self, state):
37      """ Simulates population initialization """
38      for i in range(state.P-1):
39        solution = Solution()
40        yield hold, self, gauss(state.Ta, state.TaStdev) + gauss(
              state.Tc, state.TcStdev)
41        activate(solution, solution.evaluate(state))
42
43    def evaluate(self, state):
44      """ Simulates one slave node evaluating and returning a
            solution """
45      yield request, self, state.slave
46      begin = now()
47      yield hold, self, gauss(state.Tf, state.TfStdev)
48      yield release, self, state.slave
49      state.slaveMonitor.observe(now()-begin)
50
51      solution = Solution()
52      activate(solution, solution.process(state))
53
54    def process(self, state):
55      """ Simulates the master node receiving an evaluated solution
            and generating another """
56      global NFE
57
58      arrive = now()
59      yield request, self, state.master
60      state.masterMonitor.observe(now()-arrive)
61      yield hold, self, gauss(state.Ta, state.TaStdev) + gauss(state
            .Tc, state.TcStdev)
62      state.NFE = state.NFE + 1
63
64      if state.NFE < state.maxNFE-(state.P-2):
65        yield hold, self, gauss(state.Tc, state.TcStdev)
```

```
66          yield release, self, state.master
67          solution = Solution()
68          activate(solution, solution.evaluate(state))
69        else:
70          yield release, self, state.master
71
72  def model(runSeed=1337, P=2, MaxNFE=10000, Ta=1, Tc=1, Tf=1000):
73    seed(runSeed)
74    master = Resource(capacity=1, name="Master")
75    slave = Resource(capacity=(P-1), name="Slave", monitored=True,
          monitorType=Monitor)
76    masterMonitor = Monitor()
77    slaveMonitor = Monitor()
78
79    initialize()
80    start = now()
81    state = State(master, slave, masterMonitor, slaveMonitor, P,
          MaxNFE, Ta, Tc, Tf)
82
83    solution = Solution()
84    activate(solution, solution.generate(state), at=0)
85    simulate(until=MaxNFE*Tf*100)
86
87    return state
```

Bibliography

Adra, S. F. and Fleming, P. J. (2009). A Diversity Management Operator for Evolutionary Many-Objective Optimisation. In *Evolutionary Multi-Criterion Optimization (EMO 2009)*, pages 81–94, Nantes, France.

Aguirre, H. and Tanaka, K. (2009). Space Partitioning with Adaptive ϵ-Ranking and Substitute Distance Assignments: A Comparative Study on Many-Objective MNK-Landscapes. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009)*, pages 547–554, Montreal, Canada.

Anderson, T. L. and Hill, P. J. (1997). *Water Marketing: The Next Generation*. Rowman and Littlefield, Lanham, MD.

Archier, G., Saltelli, A., and Sobol, I. (1997). Sensitivity Measures, ANOVA-Like Techniques and the Use of Bootstrap. *Journal of Statistical Computation and Simulation*, 58:99–120.

Bäck, T. (1994). Selective Pressure in Evolutionary Algorithms: A Characterization of Selection Mechanisms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 57–62.

Bäck, T., Fogel, D. B., and Michalewicz, Z. (1997). *Handbook of Evolutionary Computation*. Taylor & Francis, New York, NY.

Baker, G. L. and Gollub, J. P. (1990). *Chaotic Dynamics : An Introduction*. Cambridge University Press, Cambridge, England.

Balling, R. J., Taber, J. T., Brown, M. R., and Day, K. (1999). Multiobjective Urban Planning Using Genetic Algorithms. *Journal of Urban Planning and Development*, 125(2):86–99.

Bethke, A. D. (1976). Comparison of Genetic Algorithms and Gradient-Based Optimizers on Parallel Processors: Efficiency of Use of Processing Capacity. Technical Report No. 197, Logic of Computers Group, University of Michigan, Ann Arbor, MI.

Beume, N. and Rudolph, G. (2006). Faster S-Metric Calculation by Considering Dominated Hypervolume as Klee's Measure Problem. In *Second International Association of Science and Technology for Development (IASTED) Conference on Computational Intelligence*, pages 231–236, San Francisco, CA.

Bloebaum, C. L. and McGowan, A. M. R. (2010). Design of Compex Engineered Systems. *Journal of Mechanical Design*, 132(12):1–2.

Bosman, P. A. and Thierens, D. (2003). The Balance Between Proximity and Diversity in Multiobjective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 7(2):174–188.

Brekke, L. D., Maurer, E. P., Anderson, J. D., Dettinger, M. D., Townsley, E. S., Harrison, A., and Pruitt, T. (2009). Assessing Reservior Operations Risk Under Climate Change. *Water Resources Research*, 45(4).

Brill, E. D., Flach, J., Hopkins, L., and Ranjithan, S. (1990). MGA: A Decision Support System for Complex, Incompletely Defined Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(4):745–757.

Cantú-Paz, E. (1997). Designing Efficient Master-Slave Parallel Genetic Algorithms. IlliGAL Report No. 97004, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana-Champaign, IL.

Cantú-Paz, E. (1998). A Survey of Parallel Genetic Algorithms. *Calculateurs Paralleles Reseaux et Systems Repartis*, 10(2):141–171.

Cantú-Paz, E. (2000). *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA.

Cantú-Paz, E. and Goldberg, D. E. (1997a). Modeling Idealized Bounding Cases of Parallel Genetic Algorithms. In *Proceedings of the Second Annual Conference of Genetic Programming*, pages 353–361, Stanford, CA.

Cantú-Paz, E. and Goldberg, D. E. (1997b). Predicting Speedups of Idealized Bounding Cases of Parallel Genetic Algorithms. In *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 1997)*, pages 113–121, East Lansing, MI.

Characklis, G., Kirsch, B. R., Ramsey, J., Dillard, K., and Kelley, C. T. (2006). Developing Portfolios of Water Supply Transfers. *Water Resources Research*, 42(5).

Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer Science+Business Media LLC, New York, NY.

Corne, D. W. and Knowles, J. D. (2000). The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 839–848, Paris, France.

Corne, D. W. and Knowles, J. D. (2007). Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, pages 773–780, London, England.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Ltd., West Sussex, UK.

Deb, K. and Agrawal, R. B. (1994). Simulated Binary Crossover for Continuous Search Space. Technical Report IITK/ME/SMD-94027, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India.

Deb, K. and Jain, S. (2002). Running Performance Metrics for Evolutionary Multi-Objective Optimization. KanGAL Report No. 2002004, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology, Kanpur, India.

Deb, K., Joshi, D., and Anand, A. (2002a). Real-Coded Evolutionary Algorithms with Parent-Centric Recombination. *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, 1:61–66.

Deb, K., Mohan, M., and Mishra, S. (2003). A Fast Multi-objective Evolutionary Algorithm for Finding Well-Spread Pareto-Optimal Solutions. KanGAL Report No. 2003002, Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology, Kanpur, India.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2000). A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deb, K. and Saxena, D. K. (2006). Searching for Pareto-Optimal Solutions through Dimensionality Reduction for Certain Large-Dimensional Multi-Objective Optimization Problems. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 3353–3360, Vancouver, Canada.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2001). Scalable Test Problems for Evolutionary Multi-Objective Optimization. TIK-Technical Report No. 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH).

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002b). Scalable Multi-Objective Optimization Test Problems. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 825–830, Honolulu, HI.

di Pierro, F. (2006). *Many-Objective Evolutionary Algorithms and Applications to Water Resources Engineering*. PhD thesis, School of Engineering, Computer Science and Mathematics, University of Exeter, Exeter, UK.

di Pierro, F., Khu, S.-T., and Savic, D. A. (2007). An Investigation on Preference Order Ranking Scheme for Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 11(1):17–45.

Drechsler, N., Drechsler, R., and Becker, B. (2001). Multi-Objective Optimisation Based on Relation Favour. In *Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 154–166, Zurich, Switzerland.

D'Souza, B. and Simpson, T. W. (2003). A Genetic Algorithm Based Method for Product Family Design Optimization. *Engineering Optimization*, 35(1):1–18.

Edwards, A. L. (1993). *An Introduction to Linear Regression and Correlation*. W. H. Freeman and Co, San Francisco, CA.

Farina, M. and Amato, P. (2004). A Fuzzy Definition of "Optimality" for Many-Criteria Optimization Problems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 34(3):315–326.

Ferringer, M. P., Spencer, D. B., and Reed, P. (2009). Many-Objective Reconfiguration of Operational Satellite Constellations with the Large-Cluster Epsilon Non-dominated Sorting Genetic Algorithm-II. In *IEEE Congress on Evolutionary Computation (CEC 2009)*, pages 340–349, Trondheim, Norway.

Fleming, P. J., Purshouse, R. C., and Lygoe, R. J. (2005). Many-Objective Optimization: An Engineering Design Perspective. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 14–32. Guanajuato, Mexico.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference Genetic Algorithms (ICGA 1993)*, pages 416–423, Urbana-Champaign, IL.

Fonseca, C. M. and Fleming, P. J. (1996). On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In *Parallel Problem Solving from Nature (PPSN IV)*, pages 584–593, Berlin, Germany.

Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms — Part 1: A Unified Formulation. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 28(1):26–37.

Franssen, M. (2005). Arrow's Theorem, Multi-Criteria Decision Problems and Multi-Attribute Preferences in Engineering Design. *Research in Engineering Design*, 16(1):42–56.

Frederick, K. D. and Schwarz, G. (1999). Socioeconomic Impacts of Climate Change on U.S. Water Supplies. *Journal of the American Water Resources Association*, 35(6):1563–1583.

Fu, G., Kapelan, Z., Kasprzyk, J., and Reed, P. M. (2012). Optimal Design of Water Distribution Systems using Many-Objective Visual Analytics. *ASCE Journal of Water Resources Planning & Management*. In-Press.

Gettys, C. and Fisher, S. (1979). Hypothesis Plausibility and Hypothesis Generation. *Organizational Behavior and Human Performance*, 24(1):93–110.

Goh, C.-K. and Tan, K. C. (2009). *Evolutionary Multi-objective Optimization in Uncertain Environments*. Springer, Berlin, Germany.

Goldberg, D. E. (1989a). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, MA.

Goldberg, D. E. (1989b). Sizing Populations for Serial and Parallel Genetic Algorithms. In *3rd International Conference on Genetic Algorithms (ICGA 1989)*, pages 70–79, Fairfax, VA.

Goldberg, D. E. (1998). The Race, the Hurdle, and the Sweet Spot: Lessons from Genetic Algorithms for the Automation of Design Innovation and Creativity. IlliGAL Report No. 98007, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana-Champaign, IL.

Goldberg, D. E. (2002). *Design of Innovation: Lessons From and For Competent Genetic Algorithms*. Kluwer Academic Publishers, Norwell, MA.

Grassberger, P. and Procaccia, I. (1983). Measuring the Strangeness of Strange Attractors. *Physica D Nonlinear Phenomena*, 9:189–208.

Hadjigeorgalis, E. (2008). Managing Drought Through Water Markets: Farmer Preferences in the Rio Grande Basin. *Journal of the American Water Resources Association*, 44(3):594–605.

Hadka, D., Madduri, K., and Reed, P. (2013). Scalability Analysis of the Asynchronous, Master-Slave Borg Multiobjective Evolutionary Algorithm. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS), Nature Inspired Distributed Computing Workshop (NIDISC)*, Cambridge, MA.

Hadka, D. and Reed, P. (2012a). Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework. *Evolutionary Computation*. In-Press.

Hadka, D. and Reed, P. (2012b). Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization. *Evolutionary Computation*, 20(3):423–452.

Hadka, D., Reed, P., and Simpson, T. (2012). Diagnostic Assessment of the Borg MOEA for Many-Objective Product Family Design Problems. In *WCCI 2012 World Congress on Computational Intelligence, Congress on Evolutionary Computation (CEC 2012)*, pages 986–995, Brisbane, Australia.

Hanne, T. (1999). On the Convergence of Multiobjective Evolutionary Algorithms. *European Journal of Operational Research*, 117:553–564.

Hanne, T. (2001). Global Multiobjective Optimization with Evolutionary Algorithms: Selection Mechanisms and Mutation Control. In *Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 197–212, Zurich, Switzerland.

Harik, G. R. and Lobo, F. G. (1999). A Parameter-Less Genetic Algorithm. IlliGAL Report No. 99009, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana-Champaign, IL.

Heidelberger, P. and Trivedi, K. S. (1982). Queueing Network Models for Parallel Processing with Asynchronous Tasks. In *IEEE Transactions on Computers*, volume C-31, pages 1099–1109.

Hogarth, R. (1981). Beyond Discrete Biases: Functional and Dysfunctional Aspects of Judgemental Heuristics. *Psychological Bulletin*, 90(1):197–217.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Horn, J. (1995). *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois, Urbana-Champaign, Illinois.

Horn, J. and Nafpliotis, N. (1993). Multiobjective Optimization Using the Niched Pareto Genetic Algorithm. IlliGAL Report No. 93005, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana-Champaign, IL.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

Hughes, E. J. (2005). Evolutionary Many-Objective Optimisation: Many Once or One Many? In *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 222–227, Edinburgh, UK.

Ikeda, K., Kita, H., and Kobayashi, S. (2001). Failure of Pareto-Based MOEAs: Does Non-Dominated Really Mean Near Optimal? In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 957–962, Seoul, South Korea.

Iorio, A. and Li, X. (2008). Improving the Performance and Scalability of Differential Evolution. In *Proceedings of the 7th International Conference on Simulated Evolution and Learning*, pages 131–140, Melbourne, Australia.

Ishibuchi, H., Sakane, Y., Tsukamoto, N., and Nojima, Y. (2009). Adaptation of Scalarizing Functions in MOEA/D: An Adaptive Scalarizing Function-Based Multiobjective Evolutionary Algorithm. In *Evolutionary Multi-Criterion Optimization (EMO 2009)*, pages 438–452, Nantes, France.

Ishibuchi, H., Tsukamoto, N., Hitotsuyanagi, Y., and Nojima, Y. (2008a). Effectiveness of Scalability Improvement Attempts on the Performance of NSGA-II for Many-Objective Problems. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 649–656, Atlanta, GA.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008b). Behavior of Evolutionary Many-Objective Optimization. In *Tenth International Conference on Computer Modeling and Simulation (UKSIM 2008)*, pages 266–271, Cambridge, England.

Ishibuchi, H., Tsukamoto, N., Sakane, Y., and Nojima, Y. (2010). Indicator-Based Evolutionary Algorithm with Hypervolume Approximation by Achievement Scalarizing Functions. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010)*, pages 527–534, Portland, OR.

Israel, M. and Lund, J. R. (1995). Recent California Water Transfers: Implications for Water Management. *Natural Resources Journal*, 35(1):1–32.

Jenkins, M. W. and Lund, J. R. (2000). Integrating Yield and Shortage Management Under Multiple Uncertainties. *Journal of the American Water Resources Association*, 126(5):288–297.

Kasprzyk, J., Nataraj, S., Reed, P. M., and Lempert, R. J. (2012). Many-Objective Robust Decision Making for Complex Environmental Systems Undergoing Change. *Environmental Modelling & Software*, 42:55–71.

Kasprzyk, J., Reed, P., Kirsch, B., and Characklis, G. (2011). Many-Objective de Novo Water Supply Portfolio Planning Under Deep Uncertainty. *Environmental Modelling & Software*, 34:87–104.

Kasprzyk, J. R., Reed, P. M., Kirsch, B. R., and Characklis, G. W. (2009). Managing Population and Drought Risks using Many-Objective Water Portfolio Planning Under Uncertainty. *Water Resources Research*, 45(12).

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia.

Khare, V., Yao, X., and Deb, K. (2003). Performance Scaling of Multi-Objective Evolutionary Algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 376–390, Faro, Portugal.

Kirsch, B. R., Characklis, G. W., Dillard, K., and Kelley, C. T. (2009). More Efficient Optimization of Long-Term Water Supply Portfolios. *Water Resources Research*, 45(3).

Kita, H., Ono, I., and Kobayashi, S. (1999). Multi-Parental Extension of the Unimodal Normal Distribution Crossover for Real-Coded Genetic Algorithms. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC 1999)*, pages 1581–1588, Washington, DC.

Knowles, J. and Corne, D. (2002). On Metrics for Comparing Non-Dominated Sets. In *Congress on Evolutionary Computation (CEC 2002)*, pages 711–716, Honolulu, HI.

Knowles, J. and Corne, D. (2007). Quantifying the Effects of Objective Space Dimension in Evolutionary Multiobjective Optimization. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 757–771, Matsushima, Japan.

Knowles, J. D. and Corne, D. W. (1999). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8:149–172.

Kollat, J., Reed, P., and Maxwell, R. (2011). Many-Objective Groundwater Monitoring Network Design using Bias-Aware Ensemble Kalman Filtering, Evolutionary Optimization, and Visual Analytics. *Water Resources Research*, 47(2).

Kollat, J. B. and Reed, P. M. (2006). Comparison of Multi-Objective Evolutionary Algorithms for Long-Term Monitoring Design. *Advances in Water Resources*, 29(6):792–807.

Kollat, J. B. and Reed, P. M. (2007). A Computational Scaling Analysis of Multiobjective Evolutionary Algorithms in Long-Term Groundwater Monitoring Applications. *Advances in Water Resources*, 30(3):408–419.

Kukkonen, S. and Lampinen, J. (2005). GDE3: The Third Evolution Step of Generalized Differential Evolution. In *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 443–450, Guanajuato, Mexico.

Kundzewicz, Z., Mata, L., Arnell, N., Doll, P., Kabat, P., Jimenez, B., Miller, K. A., Oki, T., Sen, Z., and Shiklomanov, I. (2007). Freshwater Resources and their Management. In *Climate Change 2007: Impacts, Adaptation and Vulnerability*, pages 173–210, Cambridge, England. Cambridge University Press.

Lane, M. E., Kirshen, P. H., and Vogel, R. M. (1999). Indicators of Impacts of Global Climate Change on U.S. Water Resources. *Journal of Water Resources Planning and Management*, 125(4):194–204.

Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002). Combining Convergence and Diversity in Evolutionary Multi-Objective Optimization. *Evolutionary Computation*, 10(3):263–282.

Lund, J. R. (1995). Derived Estimation of Willingness to Pay to Avoid Probabilistic Shortage. *Water Resources Research*, 31(5):1367–1372.

Macdonald, I. (2009). Comparison of Sampling Techniques on the Performance of Monte-Carlo based Sensitivity Analysis. In *11th International IBPSA Conference (Building Simulation 2009)*, pages 992–999, Glasgow, Scotland.

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms*. PhD thesis, Department of Computer Science, University of Illinois, Urbana-Champaign, IL.

McKay, M. D., Beckman, R. J., and Conover, W. J. (1979). A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245.

Miettinen, K. M. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Norwell, MA.

Milly, P. C. D., Betancourt, J., Falkenmark, M., Hirsch, R., Kundzewicz, Z. W., Lettenmaier, D. P., and Stouffer, R. J. (2008). Stationarity is Dead: Whither Water Management? *Science*, 319:573–574.

Moral, R. J., Sahoo, D., and Dulikravich, G. S. (2006). Multi-Objective Hybrid Evolutionary Optimization with Automatic Switching. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, VA. AIAA Paper AIAA-2006-6976.

Nayfeh, A. H. and Balachandran, B. (1995). *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods*. Wiley Inter-Science, New York, NY.

Nicklow, J., Reed, P., Savic, D., Dessalegne, T., Harrell, L., Chan-Hilton, A., Karamouz, M., Minsker, B., Ostfeld, A., Singh, A., and Zechman, E. (2010). State of the Art for Genetic Algorithms and Beyond in Water Resources Planning and Management. *Journal of Water Resources Planning and Management*, 136:412–432.

Praditwong, K. and Yao, X. (2007). How Well do Multi-Objective Evolutionary Algorithms Scale to Large Problems. In *Congress on Evolutionary Computation (CEC 2007)*, pages 3959–3966, Singapore.

Purshouse, R. C. and Fleming, P. J. (2003). Evolutionary Many-Objective Optimisation: An Exploratory Analysis. In *Congress on Evolutionary Computation (CEC 2003)*, pages 2066–2073, Canberra, Australia.

Purshouse, R. C. and Fleming, P. J. (2007). On the Evolutionary Optimization of Many Conflicting Objectives. *IEEE Transactions on Evolutionary Computation*, 11(6):770–784.

Reed, P. M., Hadka, D. M., Herman, J. D., Kasprzyk, J. R., and Kollat, J. B. (2012). Evolutionary Multiobjective Optimization in Water Resources: The Past, Present, and Future. *Advances in Water Resources*, 51:438–456.

Reed, P. M., Kollat, J. B., Ferringer, M. P., and Thompson, T. G. (2008). Parallel Evolutionary Multi-Objective Optimization on Large, Heterogeneous Clusters: An Applications Perspective. *Journal of Aerospace Computing, Information, and Communication*, 5(11):460–478.

Rudolph, G. (1998). Evolutionary Search for Minimal Elements in Partially Ordered Sets. In *Proceedings of the 7th Annual Conference on Evolutionary Programming*, pages 345–353, San Diego, CA.

Rudolph, G. and Agapie, A. (2000). Convergence Properties of Some Multi-Objective Evolutionary Algorithms. In *Congress on Evolutionary Computation (CEC 2000)*, pages 1010–1016, San Diego, CA.

Saltelli, A. (2002). Making Best Use of Model Evaluations to Compute Sensitivity Indices. *Computer Physics Communications*, 145:280–297.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, Ltd., West Sussex, UK.

Saxena, D. and Deb, K. (2008). Dimensionality Reduction of Objectives and Constraints in Multi-Objective Optimization Problems: A System Design Perspective. In *IEEE Congress on Evolutionary Computation (CEC 2008)*, pages 3204 –3211, Hong Kong.

Schaffer, D. J. (1984). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN.

Schaffer, D. J., Caruana, R. A., Eshelman, L. J., and Das, R. (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In *Proceedings of the Third International Conference on Genetic Algorithms (ICGA 1989)*, pages 51–60, Fairfax, VA.

Shah, R., Reed, P. M., and Simpson, T. (2011). Many-Objective Evolutionary Optimization and Visual Analytics for Product Family Design. In Wang, L., Ng, A. H. C., and Deb, K., editors, *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 137–159. Springer, London, England.

Sheskin, D. J. (2004). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC, Boca Raton, FL.

Sierra, M. R. and Coello Coello, C. A. (2005). Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ϵ-Dominance. In *Evolutionary Multi-Criterion Optimization (EMO 2005)*, pages 505–519, Guanajuato, Mexico.

Simpson, T. W. (2005). Methods for Optimizing Product Platforms and Product Families: Overview and Classification. In *Product Platform and Product Family Design: Methods and Applications*, pages 133–156. Springer, New York, NY.

Simpson, T. W., Chen, W., Allen, J. K., and Mistree, F. (1996). Conceptual Design of a Family of Products Through the Use of the Robust Concept Exploration Method. In *6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, volume 2, pages 1535–1545, Bellevue, WA.

Simpson, T. W. and D'Souza, B. (2004). Assessing Variable Levels of Platform Commonality within a Product Family Using a Multiobjective Genetic Algorithm. *Concurrent Engineering: Research and Applications*, 12(2):119–130.

Simpson, T. W. and Martins, J. R. R. A. (2011). Multidisciplinary Design Optimization for Complex Engineered Systems: Report from a National Science Foundation Workshop. *ASME Journal of Mechanical Design*, 133(10):101002.

Sobol', I. M. (2001). Global Sensitivity Indices for Nonlinear Mathematical Models and their Monte Carlo Estimates. *Mathematics and Computers in Simulation*, 55:271–280.

Sobol', I. M. and Kucherenko, S. S. (2005). Global Sensitivity Indices for Nonlinear Mathematical Models. *Wilmott Magazine*, 2:1–6.

Srinivas, N. and Deb, K. (1994). Multiobjective Optimization using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248.

Srivastava, R. P. (2002). Time Continuation in Genetic Algorithms. IlliGAL Report No. 2001021, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana-Champaign, IL.

Storn, R. and Price, K. (1997). Differential Evolution — A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.

Sülflow, A., Drechsler, N., and Drechsler, R. (2007). Robust Multi-Objective Optimization in High Dimensional Spaces. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 715–726, Matsushima, Japan.

Tang, Y., Reed, P., and Wagener, T. (2006). How Effective and Efficient are Multiobjective Evolutionary Algorithms at Hydrologic Model Calibration? *Hydrology and Earth System Science*, 10:289–307.

Tang, Y., Reed, P., Wagener, T., and van Werkhoven, K. (2007). Comparing Sensitivity Analysis Methods to Advance Lumped Watershed Model Identification and Evaluation. *Hydrology and Earth System Sciences*, 11(2):793–817.

Teytaud, O. (2006). How Entropy-Theorems can show that On-line Approximating High-Dim Pareto-Fronts is too Hard. In *International Conference on Parallel Problem Solving from Nature (PPSN), Bridging the Gap between Theory and Practice (BTP) Workshop*.

Teytaud, O. (2007). On the Hardness of Offline Multi-Objective Optimization. *Evolutionary Computation*, 15(4):475–491.

Tsutsui, S., Yamamura, M., and Higuchi, T. (1999). Multi-Parent Recombination with Simplex Crossover in Real Coded Genetic Algorithms. In *Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 657–664, Orlando, FL.

Venkataraman, S. and Haftka, R. T. (2004). Structural Optimization Complexity: What Has Moore's Law Done for Us? *Structural and Multidisciplinary Optimization*, 28(6):375–387.

Vorosmarty, C. J., Green, P., Salisbury, J., and Lammers, R. B. (2000). Global Water Resources: Vulnerability from Climate Change and Population Growth. *Science*, 289:284–288.

Vrugt, J. A. and Robinson, B. A. (2007). Improved Evolutionary Optimization from Genetically Adaptive Multimethod Search. *Proceedings of the National Academy of Sciences*, 104(3):708–711.

Vrugt, J. A., Robinson, B. A., and Hyman, J. M. (2009). Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces. *IEEE Transactions on Evolutionary Computation*, 13(2):243–259.

Wagner, T., Beume, N., and Naujoks, B. (2007). Pareto-, Aggregation-, and Indicator-Based Methods in Many-Objective Optimization. In *Evolutionary Multi-Criterion Optimization (EMO 2007)*, pages 742–756, Matsushima, Japan.

Wang, L., Ng, A. H. C., and Deb, K. (2011). *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing*. Springer-Verlag, London, England.

Watkins Jr., D. W. and McKinney, D. C. (1999). Screening Water Supply Options for the Edwards Aquifier Region in Central Texas. *Journal of Water Resources Planning and Management*, 125(1):14–24.

While, L., Bradstreet, L., and Barone, L. (2012). A Fast Way of Calculating Exact Hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95.

Wilchfort, O. and Lund, J. R. (1997). Shortage Management Modeling for Urban Water Supply Systems. *Journal of the American Water Resources Association*, 123(4):250–258.

Woodruff, M., Hadka, D., Reed, P., and Simpson, T. (2012). Auto-Adaptive Search Capabilities of the New Borg MOEA: A Detailed Comparison on Product Family Design Problems. In *12 AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Indianapolis, IN. AIAA Paper AIAA-2012-5442.

Woodruff, M., Reed, P. M., and Simpson, T. (2013). Many-Objective Visual Analytics: Rethinking the Design of Complex Engineered Systems. *Structural and Multidisciplinary Optimization*. In-Press.

Zhang, Q., Liu, W., and Li, H. (2009a). The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances. In *Congress on Evolutionary Computation (CEC 2009)*, pages 203–208, Trondheim, Norway.

Zhang, Q. and Suganthan, P. N. (2009). Final Report on CEC'09 MOEA Competition. In *Congress on Evolutionary Computation (CEC 2009)*, Trondheim, Norway.

Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W., and Tiwari, S. (2009b). Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition. Technical Report CES-487, School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK.

Zhang, X., Srinivasan, R., and Liew, M. V. (2010). On the use of Multi-Algorithm, Genetically Adaptive Multi-Objective Method for Multi-Site Calibration of the SWAT Model. *Hydrological Processes*, 24(8):955–969.

Zitzler, E. and Künzli, S. (2004). Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, Birmingham, UK.

Zitzler, E., Laumanns, M., and Thiele, L. (2002a). *SPEA2: Improving the Strength Pareto Evolutionary Algorithm For Multiobjective Optimization*. International Center for Numerical Methods in Engineering (CIMNE), Barcelona, Spain.

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C. M., and da Fonseca, V. G. (2002b). Why Quality Assessment of Multiobjective Optimizers Is Difficult. In *Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 666–674, New York, NY.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2002c). Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132.

Vita: David M. Hadka

Education

The Pennsylvania State University — University Park, PA

- B.S., Computer Science with Mathematics minor · 2005
- Ph.D., Computer Science and Engineering · 2013

Experience

Applied Research Laboratory — State College, PA
R&D Engineer — 2005-Present

- Lead software engineer on the $48M DARPA iFAB Foundry project.
- Graduate assistant from 2005-2012 developing mission-critical tools in use by the US DoD and defense manufacturers.

Select Publications

1. *Evolutionary Computing Based Optimization.* Disclosed in utility patent application 13/356,391 and provisional patent application 61/437,846.
2. Hadka, D. et al. *Scalability Analysis of the Asynchronous, Master-Slave Borg Multi-objective Evolutionary Algorithm.* 27th International Parallel & Distributed Processing Symposium (IPDPS), Nature-Inspired Distributed Computing (NIDISC) Workshop, Boston, MA, 20-24 May 2013.
3. Hadka, D. et al. *Diagnostic Assessment of the Borg MOEA on Many-Objective Product Family Design Problems.* WCCI 2012 World Congress on Computational Intelligence, Congress on Evolutionary Computation, Brisbane, Australia, 10-15 June 2012, pp. 986-995.
4. Reed, P., et al. *Evolutionary Multiobjective Optimization in Water Resources: The Past, Present & Future.* Advances in Water Resources (Editor Invited Submission to the 35th Anniversary Special Issue), 2012.
5. Hadka, D. and Reed, P. *Diagnostic Assessment of Search Controls and Failure Modes in Many-Objective Evolutionary Optimization.* Evolutionary Computation, 20(3):423-452, 2012.
6. Hadka, D. and Reed, P. *Borg: An Auto-Adaptive Many-Objective Evolutionary Computing Framework.* Evolutionary Computation, 2012.